# MECHATRONICS

## AN INTRODUCTION

Edited by

Robert H. Bishop

# MECHATRONICS

## AN INTRODUCTION

# MECHATRONICS

## AN INTRODUCTION

Robert H. Bishop
University of Texas at Austin
U.S.A.


Taylor & Francis
Taylor & Francis Group
Boca Raton London New York

A CRC title, part of the Taylor & Francis imprint, a member of the
Taylor & Francis Group, the academic division of T&F Informa plc.

Published in 2006 by
CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742


Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number-10: 0-8493-6358-6 (Hardcover)
International Standard Book Number-13: 978-0-8493-6358-0 (Hardcover)
Library of Congress Card Number 2005049656



**Library of Congress Cataloging-in-Publication Data**

Mechatronics : an introduction / edited by Robert H. Bishop.
p. cm.
ISBN 0-8493-6358-6 (alk. paper)
1. Mechatronics. I. Bishop, Robert H., 1957-

TJ163.12.M4315 2005
621.3--dc22 2005049656

T&F informa
Taylor & Francis Group
is the Academic Division of T&F Informa plc.

**Visit the Taylor & Francis Web site at**
**http://www.taylorandfrancis.com**

**and the CRC Press Web site at**
**http://www.crcpress.com**

# Preface

According to the original definition of mechatronics that the Yasakawa Electric Company proposed and the definitions that have since appeared, many engineering products designed and manufactured in the last thirty years that integrate mechanical and electrical systems can be classified as *mechatronic systems*. In trademark application documents, Yasakawa defined mechatronics in this way:

> The word *mechatronics* is composed of "mecha" from mechanism and the "tronics" from electronics. In other words, technologies and developed products will be incorporating electronics more and more intimately and organically into mechanisms, making it impossible to tell where one ends and the other begins.

Where is mechatronics today? The advent of the microcomputer, embedded computers, and associated information technologies and software advances have led to important advances in mechatronics. For example, consider the automobile. In the early stages of automobile design, the radio was the only significant electronics in it. All other functions were entirely mechanical or electrical. Today, there are about 30–60 microcontrollers in a car. And with the drive to develop modular systems for plug-n-play mechatronics subsystems, this is expected to increase.

*Mechatronics: An Introduction* provides an introduction to the vibrant field of mechatronics. As the historical divisions between the various branches of engineering and computer science become less clearly defined, the mechatronics specialty provides a roadmap for nontraditional engineering students studying within the traditional structure of most engineering colleges. Evidently, mechatronics laboratories and classes in the university environment are expanding world-wide. The list of contributors to this book that includes authors from around the globe reflects this.

The material in *Mechatronics: An Introduction* appeared in a more complete form in *The Mechatronics Handbook* that CRC Press and ISA-The Instrumentation, Systems, and Automation Society copublished. The *Mechatronics Handbook* was conceived as a reference resource for research and development departments in academia, government, and industry, and for university libraries. It also was intended as a resource for scholars interested in understanding and explaining the engineering design process. The success of the full-scale handbook spawned the idea that a more condensed book, providing a general impression of the subject, would benefit those searching for an overview of the mechatronics. This book intends to serve this new audience.

## Organization

*Mechatronics: An Introduction* is a collection of 21 articles covering the key elements of mechatronics:

a. Physical Systems Modeling
b. Sensors and Actuators
c. Signals and Systems

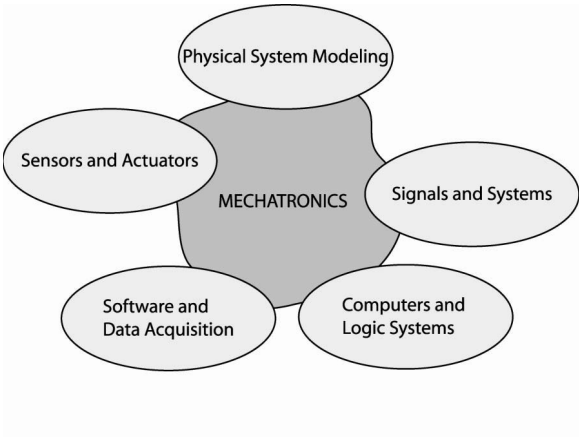



**FIGURE 1** Key Elements of Mechatronics

d. Computers and Logic Systems
e. Software and Data Acquisition

The opening five articles define and organize mechatronics. These articles constitute an overview introducing the key elements of mechatronics. Listed in order of appearance, the articles are:

- What is Mechatronics?
- Mechatronic Design Approach
- System Interfacing, Instrumentation, and Control Systems
- Microprocessor-Based Controllers and Microelectronics
- An Introduction to Micro- and Nanotechnology

One of the main elements of mechatronics is physical system modeling. The next three articles discuss an overview of the underlying mechanical and electrical mathematical models comprising most mechatronic systems, including the important topics of microelectromechanical systems (MEMS) and traditional electro-mechanical systems. Listed in order of appearance, the articles are:

- Modeling Electromechanical Systems
- Modeling and Simulation for MEMS
- The Physical Basis of Analogies in Physical System Models

The next three articles summarize the essential elements of sensors and actuators for mechatronics. This section begins with an introduction to the subject and concludes with articles on the important subjects of time and frequency, and on sensor and actuator characteristics. Listed in order of appearance, the articles are:

- Introduction to Sensors and Actuators
- Fundamentals of Time and Frequency
- Sensor and Actuator Characteristics

Signals and systems are key elements of any mechatronic system. Control systems and other subsystems that comprise "smart products" are included in this general area of study. Since significant material on the general subject of signals and systems is readily available to the reader, that material is not repeated here. Instead, the next group of articles presents the relevant aspects of signals and systems of special importance to the study of mechatronics. The articles describe the role of control in mechatronics and the role of modeling in mechatronic design, and conclude with a discussion of design optimization. Listed in order of appearance, the three articles are:

- The Role of Controls in Mechatronics
- The Role of Modeling in Mechatronics Design
- Design Optimization of Mechatronic Systems

The development of the computer has profoundly impacted the world. This is especially true in mechatronics, where the integration of computers with electromechanical systems has led to a new generation of smart products. The future is filled with promise of better and more intelligent products, resulting from continued improvements in computer technology and software engineering. The last seven articles of the book are devoted to the topics of computers and software. The next four articles focus on computer hardware and associated issues of logic, communication, networking, interfacing, embedded computers, and programmable logic controllers. Listed in order of appearance, the articles are:

- Introduction to Computers and Logic Systems
- System Interfaces
- Communication and Computer Networks
- Control with Embedded Computers and Programmable Logic Controllers

Since computers play a central role in modern mechatronics products, it is important to understand how data is acquired and makes its way into the computer for processing and logging. The final three articles focus on issues surrounding computer software and data acquisition. Listed in order of appearance, the articles are:

- Introduction to Data Acquisition
- Computer-Based Instrumentation Systems
- Software Design and Development

## Acknowledgments

I wish to express my heartfelt thanks to all the contributing authors. I appreciate their taking time in otherwise busy and hectic schedules to author the excellent articles appearing in *Mechatronics: An Introduction*. I also wish to thank my advisory board for their help in the development of *The Mechatronics Handbook*, the basis of the articles in this volume.

This book is a result of a collaborative effort that CRC Press expertly managed. My thanks to the editorial and production staff:

Nora Konopka, Acquisitions Editor
Michael Buso, Project Coordinator
Susan Fox, Project Editor

Thanks to my friend and collaborator Professor Richard C. Dorf for his continued support and guidance. And finally, a special thanks to Lynda Bishop for managing the incoming and outgoing draft manuscripts. Her organizational skills were invaluable to this project.

**Robert H. Bishop**
Editor-in-Chief

# Editor-in-Chief

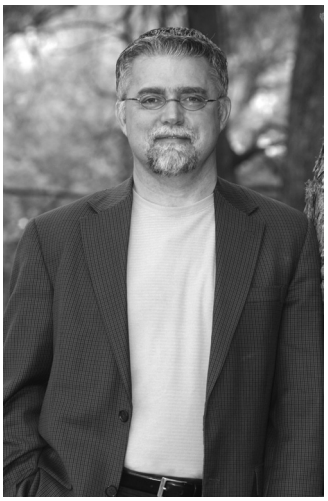


Photo courtesy of Caroling Lee

**Robert H. Bishop** is a professor of aerospace engineering and engineering mechanics at The University of Texas at Austin and holds the Myron L. Begeman Fellowship in Engineering. He received his B.S. and M.S. degrees from Texas A&M University in aerospace engineering, and his Ph.D. from Rice University in electrical and computer engineering. Prior to joining The University of Texas at Austin, he was a member of the technical staff at the MIT Charles Stark Draper Laboratory. Dr. Bishop is a specialist in the area of planetary exploration with an emphasis on spacecraft guidance, navigation, and control. He is currently working with NASA Johnson Space Center and the Jet Propulsion Laboratory on techniques for achieving precision landing on Mars. He is an active researcher, authoring and co-authoring over 50 journal and conference papers. The Boeing Company selected him two times as a Faculty Fellow at the NASA Jet Propulsion Laboratory and a Welliver Faculty Fellow. Dr. Bishop co-authored *Modern Control Systems* with Prof. R. C. Dorf and he authored two other books, entitled *Learning with LabView* and *Modern Control System Design and Analysis Using Matlab and Simulink*. He recently received the John Leland Atwood Award from the American Society of Engineering Educators and the American Institute of Aeronautics and Astronautics that is periodically given to "a leader who has made lasting and significant contributions to aerospace engineering education."

# Contributors


**M. Anjanappa**
University of Maryland
Baltimore, Maryland

**Eric J. Barth**
Vanderbilt University
Nashville, Tennessee

**Peter Breedveld**
University of Twente
Enschede, The Netherlands

**Tomas Brezina**
Technical University of Brno
Brno, Czech Republic

**Kevin C. Craig**
Rensselaer Polytechnic Institute
Troy, New York

**Jace Curtis**
National Instruments, Inc.
Austin, Texas

**K. Datta**
University of Maryland
Baltimore, Maryland

**Ivan Dolezal**
Technical University of Liberec
Liberec, Czech Republic

**Kris Fuller**
National Instruments, Inc.
Austin, Texas

**Michael Goldfarb**
Vanderbilt University
Nashville, Tennessee

**Margaret H. Hamilton**
Hamilton Technologies, Inc.
Cambridge, Massachusetts

**Neville Hogan**
Massachusetts Institute of Technology
Cambridge, Massachusetts

**Rick Homkes**
Purdue University
Kokomo, Indiana

**Mohammad Ilyas**
Florida Atlantic University
Boca Raton, Florida

**Rolf Isermann**
Darmstadt University of Technology
Darmstadt, Germany

**Hugh Jack**
Grand Valley State University
Grand Rapids, Michigan

**Jeffrey A. Jalkio**
University of St. Thomas
St. Paul, Minnesota

**J. Katupitiya**
The University of New South Wales
Sydney, Australia

**Ctirad Kratochvil**
Technical University of Brno
Brno, Czech Republic

**Michael A. Lombardi**
National Institute of Standards and Technology
Boulder, Colorado

**Francis C. Moon**
Cornell University
Ithaca, New York

**Ondrej Novak**
Technical University of Liberec
Liberec, Czech Republic

**Cestmir Ondrusek**
Technical University of Brno
Brno, Czech Republic

**Joey Parker**
University of Alabama
Tuscaloosa, Alabama

**Carla Purdy**
University of Cincinnati
Cincinnati, Ohio

**M. K. Ramasubramanian**
North Carolina State University
Raleigh, North Carolina

**T. Song**
University of Maryland
Baltimore, Maryland

**Andrew Sterian**
Grand Valley State University
Grand Rapids, Michigan

**Alvin Strauss**
Vanderbilt University
Nashville, Tennessee

**Fred Stolfi**
Rensselaer Polytechnic Institute
Troy, New York

**M. J. Tordon**
The University of New South Wales
Sydney, Australia

**Job van Amerongen**
University of Twente
Enschede, The Netherlands

# Contents

# 1

# What is Mechatronics?

Robert H. Bishop
*The University of Texas at Austin*

M. K. Ramasubramanian
*North Carolina State University*



Mechatronics is a natural stage in the evolutionary process of modern engineering design. The development of the computer, and then the microcomputer, embedded computers, and associated information technologies and software advances, made mechatronics an imperative in the latter part of the twentieth century. Standing at the threshold of the twenty-first century, with expected advances in integrated bio-electro-mechanical systems, quantum computers, nano- and pico-systems, and other unforeseen developments, the future of mechatronics is full of potential and bright possibilities.

## 1.1 Basic Definitions

The definition of mechatronics has evolved since the original definition by the Yasakawa Electric Company. In trademark application documents, Yasakawa defined mechatronics in this way [1,2]:

> The word, mechatronics, is composed of "mecha" from mechanism and the "tronics" from electronics. In other words, technologies and developed products will be incorporating electronics more and more into mechanisms, intimately and organically, and making it impossible to tell where one ends and the other begins.

The definition of mechatronics continued to evolve after Yasakawa suggested the original definition. One oft quoted definition of mechatronics was presented by Harashima, Tomizuka, and Fukada in 1996 [3]. In their words, mechatronics is defined as

> the synergistic integration of mechanical engineering, with electronics and intelligent computer control in the design and manufacturing of industrial products and processes.

That same year, another definition was suggested by Auslander and Kempf [4]:

> Mechatronics is the application of complex decision making to the operation of physical systems.

Yet another definition due to Shetty and Kolk appeared in 1997 [5]:

> Mechatronics is a methodology used for the optimal design of electromechanical products.

More recently, we find the suggestion by W. Bolton [6]:

> A mechatronic system is not just a marriage of electrical and mechanical systems and is more than just a control system; it is a complete integration of all of them.

All of these definitions and statements about mechatronics are accurate and informative, yet each one in and of itself fails to capture the totality of mechatronics. Despite continuing efforts to define mechatronics, to classify mechatronic products, and to develop a standard mechatronics curriculum, a consensus opinion on an all-encompassing description of "what is mechatronics" eludes us. This lack of consensus is a healthy sign. It says that the field is alive, that it is a youthful subject. Even without an unarguably definitive description of mechatronics, engineers understand from the definitions given above and from their own personal experiences the essence of the *philosophy* of mechatronics.

For many practicing engineers on the front line of engineering design, mechatronics is nothing new. Many engineering products of the last 25 years integrated mechanical, electrical, and computer systems, yet were designed by engineers that were never formally trained in mechatronics *per se*. It appears that modern concurrent engineering design practices, now formally viewed as part of the mechatronics specialty, are natural design processes. What is evident is that the study of mechatronics provides a mechanism for scholars interested in understanding and explaining the engineering design process to define, classify, organize, and integrate many aspects of product design into a coherent package. As the historical divisions between mechanical, electrical, aerospace, chemical, civil, and computer engineering become less clearly defined, we should take comfort in the existence of mechatronics as a field of study in academia. The mechatronics specialty provides an educational path, that is, a roadmap, for engineering students studying within the traditional structure of most engineering colleges. Mechatronics is generally recognized worldwide as a vibrant area of study. Undergraduate and graduate programs in mechatronic engineering are now offered in many universities. Refereed journals are being published and dedicated conferences are being organized and are generally highly attended.

It should be understood that mechatronics is not just a convenient structure for investigative studies by academicians; it is a way of life in modern engineering practice. The introduction of the microprocessor in the early 1980s and the ever increasing desired performance to cost ratio revolutionized the paradigm of engineering design. The number of new products being developed at the intersection of traditional disciplines of engineering, computer science, and the natural sciences is ever increasing. New developments in these traditional disciplines are being absorbed into mechatronics design at an ever increasing pace. The ongoing information technology revolution, advances in wireless communication, smart sensors design (enabled by MEMS technology), and embedded systems engineering ensures that the engineering design paradigm will continue to evolve in the early twenty-first century.

## 1.2 Key Elements of Mechatronics

The study of mechatronic systems can be divided into the following areas of specialty:

1. Physical Systems Modeling
2. Sensors and Actuators
3. Signals and Systems
4. Computers and Logic Systems
5. Software and Data Acquisition

The key elements of mechatronics are illustrated in Figure 1.1. As the field of mechatronics continues to mature, the list of relevant topics associated with the area will most certainly expand and evolve.

## 1.3 Historical Perspective

Attempts to construct automated mechanical systems has an interesting history. Actually, the term "automation" was not popularized until the 1940s when it was coined by the Ford Motor Company to denote a process in which a machine transferred a sub-assembly item from one station to another and then positioned the item precisely for additional assembly operations. But successful development of automated mechanical systems occurred long before then. For example, early applications of automatic control

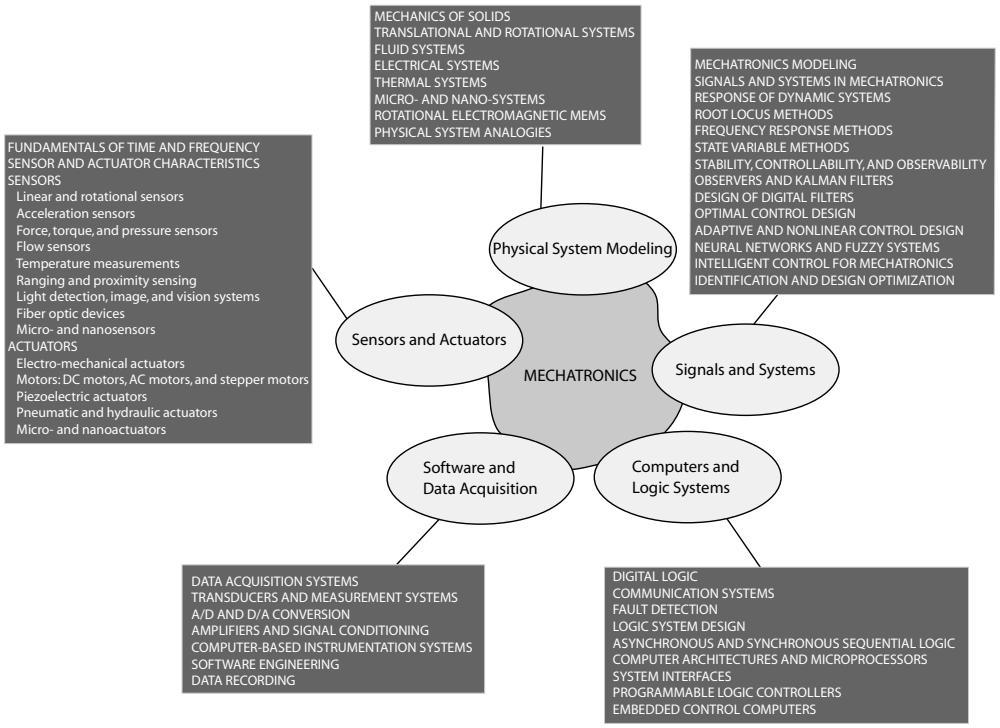



**FIGURE 1.1** The key elements of mechatronics.

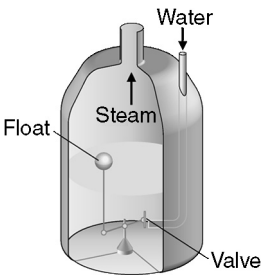



**FIGURE 1.2** Water-level float regulator. (From *Modern Control Systems*, 9th ed., R. C. Dorf and R. H. Bishop, Prentice-Hall, 2001. Used with permission.)

systems appeared in Greece from 300 to 1 B.C. with the development of float regulator mechanisms [7]. Two important examples include the water clock of Ktesibios that used a float regulator, and an oil lamp devised by Philon, which also used a float regulator to maintain a constant level of fuel oil. Later, in the first century, Heron of Alexandria published a book entitled *Pneumatica* that described different types of water-level mechanisms using float regulators.

In Europe and Russia, between seventeenth and nineteenth centuries, many important devices were invented that would eventually contribute to mechatronics. Cornelis Drebbel (1572–1633) of Holland devised the temperature regulator representing one of the first feedback systems of that era. Subsequently, Dennis Papin (1647–1712) invented a pressure safety regulator for steam boilers in 1681. Papin's pressure regulator is similar to a modern-day pressure-cooker valve. The first mechanical calculating machine was invented by Pascal in 1642 [8]. The first historical feedback system claimed by Russia was developed by Polzunov in 1765 [9]. Polzunov's water-level float regulator, illustrated in Figure 1.2, employs a float that rises and lowers in relation to the water level, thereby controlling the valve that covers the water inlet in the boiler.

Further evolution in automation was enabled by advancements in control theory traced back to the Watt flyball governor of 1769. The flyball governor, illustrated in Figure 1.3, was used to control the

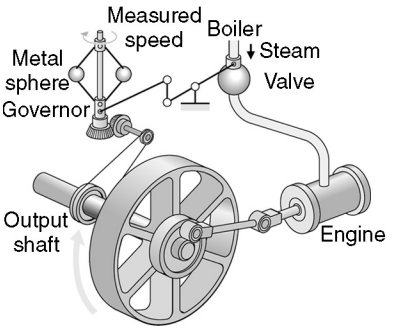



**FIGURE 1.3** Watt's flyball governor. (From *Modern Control Systems,* 9th ed., R. C. Dorf and R. H. Bishop, Prentice-Hall, 2001. Used with permission.)

speed of a steam engine [10]. Employing a measurement of the speed of the output shaft and utilizing the motion of the flyball to control the valve, the amount of steam entering the engine is controlled. As the speed of the engine increases, the metal spheres on the governor apparatus rise and extend away from the shaft axis, thereby closing the valve. This is an example of a feedback control system where the feedback signal and the control actuation are completely coupled in the mechanical hardware.

These early successful automation developments were achieved through intuition, application of practical skills, and persistence. The next step in the evolution of automation required a *theory* of automatic control. The precursor to the numerically controlled (NC) machines for automated manufacturing (to be developed in the 1950s and 60s at MIT) appeared in the early 1800s with the invention of feed-forward control of weaving looms by Joseph Jacquard of France. In the late 1800s, the subject now known as control theory was initiated by J. C. Maxwell through analysis of the set of differential equations describing the flyball governor [11]. Maxwell investigated the effect various system parameters had on the system performance. At about the same time, Vyshnegradskii formulated a mathematical theory of regulators [12]. In the 1830s, Michael Faraday described the law of induction that would form the basis of the electric motor and the electric dynamo. Subsequently, in the late 1880s, Nikola Tesla invented the alternating-current induction motor. The basic idea of controlling a mechanical system automatically was firmly established by the end of 1800s. The evolution of automation would accelerate significantly in the twentieth century.

The development of pneumatic control elements in the 1930s matured to a point of finding applications in the process industries. However, prior to 1940, the design of control systems remained an art generally characterized by trial-and-error methods. During the 1940s, continued advances in mathematical and analytical methods solidified the notion of control engineering as an independent engineering discipline. In the United States, the development of the telephone system and electronic feedback amplifiers spurred the use of feedback by Bode, Nyquist, and Black at Bell Telephone Laboratories [13–17]. The operation of the feedback amplifiers was described in the frequency domain and the ensuing design and analysis practices are now generally classified as "classical control." During the same time period, control theory was also developing in Russia and eastern Europe. Mathematicians and applied mechanicians in the former Soviet Union dominated the field of controls and concentrated on time domain formulations and differential equation models of systems. Further developments of time domain formulations using state variable system representations occurred in the 1960s and led to design and analysis practices now generally classified as "modern control."

The World War II war effort led to further advances in the theory and practice of automatic control in an effort to design and construct automatic airplane pilots, gun-positioning systems, radar antenna control systems, and other military systems. The complexity and expected performance of these military systems necessitated an extension of the available control techniques and fostered interest in control systems and the development of new insights and methods. Frequency domain techniques continued to dominate the field of controls following World War II, with the increased use of the Laplace transform, and the use of the so-called *s*-plane methods, such as designing control systems using root locus.

On the commercial side, driven by cost savings achieved through mass production, automation of the production process was a high priority beginning in the 1940s. During the 1950s, the invention of the cam, linkages, and chain drives became the major enabling technologies for the invention of new products and high-speed precision manufacturing and assembly. Examples include textile and printing machines, paper converting machinery, and sewing machines. High-volume precision manufacturing became a reality during this period. The automated paperboard container-manufacturing machine employs a sheet-fed process wherein the paperboard is cut into a fan shape to form the tapered sidewall, and wrapped around a mandrel. The seam is then heat sealed and held until cured. Another sheet-fed source of paperboard is used to cut out the plate to form the bottom of the paperboard container, formed into a shallow dish through scoring and creasing operations in a die, and assembled to the cup shell. The lower edge of the cup shell is bent inwards over the edge of the bottom plate sidewall, and heat-sealed under high pressure to prevent leaks and provide a precisely level edge for standup. The brim is formed on the top to provide a ring-on-shell structure to provide the stiffness needed for its functionality. All of these operations are carried out while the work piece undergoes a precision transfer from one turret to another and is then ejected. The production rate of a typical machine averages over 200 cups per minute. The automated paperboard container manufacturing did not involve any non-mechanical system except an electric motor for driving the line shaft. These machines are typical of paper converting and textile machinery and represent automated systems significantly more complex than their predecessors.

The development of the microprocessor in the late 1960s led to early forms of computer control in process and product design. Examples include numerically controlled (NC) machines and aircraft control systems. Yet the manufacturing processes were still entirely mechanical in nature and the automation and control systems were implemented only as an afterthought. The launch of Sputnik and the advent of the space age provided yet another impetus to the continued development of controlled mechanical systems. Missiles and space probes necessitated the development of complex, highly accurate control systems. Furthermore, the need to minimize satellite mass (that is, to minimize the amount of fuel required for the mission) while providing accurate control encouraged advancements in the important field of optimal control. Time domain methods developed by Liapunov, Minorsky, and others, as well as the theories of optimal control developed by L. S. Pontryagin in the former Soviet Union and R. Bellman in the United States, were well matched with the increasing availability of high-speed computers and new programming languages for scientific use.

Advancements in semiconductor and integrated circuits manufacturing led to the development of a new class of products that incorporated mechanical and electronics in the system and required the two together for their functionality. The term mechatronics was introduced by Yasakawa Electric in 1969 to represent such systems. Yasakawa was granted a trademark in 1972, but after widespread usage of the term, released its trademark rights in 1982 [1–3]. Initially, mechatronics referred to systems with only mechanical systems and electrical components—no computation was involved. Examples of such systems include the automatic sliding door, vending machines, and garage door openers.

In the late 1970s, the Japan Society for the Promotion of Machine Industry (JSPMI) classified mechatronics products into four categories [1]:

1. *Class I:* Primarily mechanical products with electronics incorporated to enhance functionality. Examples include numerically controlled machine tools and variable speed drives in manufacturing machines.
2. *Class II:* Traditional mechanical systems with significantly updated internal devices incorporating electronics. The external user interfaces are unaltered. Examples include the modern sewing machine and automated manufacturing systems.
3. *Class III:* Systems that retain the functionality of the traditional mechanical system, but the internal mechanisms are replaced by electronics. An example is the digital watch.
4. *Class IV:* Products designed with mechanical and electronic technologies through synergistic integration. Examples include photocopiers, intelligent washers and dryers, rice cookers, and automatic ovens.

The enabling technologies for each mechatronic product class illustrate the progression of electromechanical products in stride with developments in control theory, computation technologies, and microprocessors. Class I products were enabled by servo technology, power electronics, and control theory. Class II products were enabled by the availability of early computational and memory devices and custom circuit design capabilities. Class III products relied heavily on the microprocessor and integrated circuits to replace mechanical systems. Finally, Class IV products marked the beginning of true mechatronic systems, through integration of mechanical systems and electronics. It was not until the 1970s with the development of the microprocessor by the Intel Corporation that integration of computational systems with mechanical systems became practical.

The divide between classical control and modern control was significantly reduced in the 1980s with the advent of "robust control" theory. It is now generally accepted that control engineering must consider both the time domain and the frequency domain approaches simultaneously in the analysis and design of control systems. Also, during the 1980s, the utilization of digital computers as integral components of control systems became routine. There are literally hundreds of thousands of digital process control computers installed worldwide [18,19]. Whatever definition of mechatronics one chooses to adopt, it is evident that modern mechatronics involves computation as the central element. In fact, the incorporation of the microprocessor to precisely modulate mechanical power and to adapt to changes in environment are the essence of modern mechatronics and smart products.

## 1.4 The Development of the Automobile as a Mechatronic System

The evolution of modern mechatronics can be illustrated with the example of the automobile. Until the 1960s, the radio was the only significant electronics in an automobile. All other functions were entirely mechanical or electrical, such as the starter motor and the battery charging systems. There were no "intelligent safety systems," except augmenting the bumper and structural members to protect occupants in case of accidents. Seat belts, introduced in the early 1960s, were aimed at improving occupant safety and were completely mechanically actuated. All engine systems were controlled by the driver and/or other mechanical control systems. For instance, before the introduction of sensors and microcontrollers, a mechanical distributor was used to select the specific spark plug to fire when the fuel–air mixture was compressed. The timing of the ignition was the control variable. The mechanically controlled combustion process was not optimal in terms of fuel efficiency. Modeling of the combustion process showed that, for increased fuel efficiency, there existed an optimal time when the fuel should be ignited. The timing depends on load, speed, and other measurable quantities. The electronic ignition system was one of the first mechatronic systems to be introduced in the automobile in the late 1970s. The electronic ignition system consists of a crankshaft position sensor, camshaft position sensor, airflow rate, throttle position, rate of throttle position change sensors, and a dedicated microcontroller determining the timing of the spark plug firings. Early implementations involved only a Hall effect sensor to sense the position of the rotor in the distributor accurately. Subsequent implementations eliminated the distributor completely and directly controlled the firings utilizing a microprocessor.

The Antilock Brake System (ABS) was also introduced in the late 1970s in automobiles [20]. The ABS works by sensing lockup of any of the wheels and then modulating the hydraulic pressure as needed to minimize or eliminate sliding. The Traction Control System (TCS) was introduced in automobiles in the mid-1990s. The TCS works by sensing slippage during acceleration and then modulating the power to the slipping wheel. This process ensures that the vehicle is accelerating at the maximum possible rate under given road and vehicle conditions. The Vehicle Dynamics Control (VDC) system was introduced in automobiles in the late 1990s. The VDC works similar to the TCS with the addition of a yaw rate sensor and a lateral accelerometer. The driver intention is determined by the steering wheel position and then compared with the actual direction of motion. The TCS system is then activated to control the

power to the wheels and to control the vehicle velocity and minimize the difference between the steering wheel direction and the direction of the vehicle motion [20,21]. In some cases, the ABS is used to slow down the vehicle to achieve desired control. In automobiles today, typically, 8, 16, or 32-bit CPUs are used for implementation of the various control systems. The microcontroller has onboard memory (EEPROM/EPROM), digital and analog inputs, A/D converters, pulse width modulation (PWM), timer functions, such as event counting and pulse width measurement, prioritized inputs, and in some cases digital signal processing. The 32-bit processor is used for engine management, transmission control, and airbags; the 16-bit processor is used for the ABS, TCS, VDC, instrument cluster, and air conditioning systems; the 8-bit processor is used for seat, mirror control, and window lift systems. Today, there are about 30–60 microcontrollers in a car. This is expected to increase with the drive towards developing modular systems for plug-n-ply mechatronics subsystems.

Mechatronics has become a necessity for product differentiation in automobiles. Since the basics of internal combustion engine were worked out almost a century ago, differences in the engine design among the various automobiles are no longer useful as a product differentiator. In the 1970s, the Japanese automakers succeeded in establishing a foothold in the U.S. automobile market by offering unsurpassed quality and fuel-efficient small automobiles. The quality of the vehicle was the product differentiator through the 1980s. In the 1990s, consumers came to expect quality and reliability in automobiles from all manufacturers. Today, *mechatronic features* have become the product differentiator in these traditionally mechanical systems. This is further accelerated by higher performance price ratio in electronics, market demand for innovative products with smart features, and the drive to reduce cost of manufacturing of existing products through redesign incorporating mechatronics elements. With the prospects of low single digit (2–3%) growth, automotive makers will be searching for high-tech features that will differentiate their vehicles from others [22]. The automotive electronics market in North America, now at about $20 billion, is expected to reach $28 billion by 2004 [22]. New applications of mechatronic systems in the automotive world include semi-autonomous to fully autonomous automobiles, safety enhancements, emission reduction, and other features including intelligent cruise control, and brake by wire systems eliminating the hydraulics [23]. Another significant growth area that would benefit from a mechatronics design approach is wireless networking of automobiles to ground stations and vehicle-to-vehicle communication. Telematics, which combines audio, hands-free cell phone, navigation, Internet connectivity, e-mail, and voice recognition, is perhaps the largest potential automotive growth area. In fact, the use of electronics in automobiles is expected to increase at an annual rate of 6% per year over the next five years, and the electronics functionality will double over the next five years [24].

Micro Electromechanical Systems (MEMS) is an enabling technology for the cost-effective development of sensors and actuators for mechatronics applications. Already, several MEMS devices are in use in automobiles, including sensors and actuators for airbag deployment and pressure sensors for manifold pressure measurement. Integrating MEMS devices with CMOS signal conditioning circuits on the same silicon chip is another example of development of enabling technologies that will improve mechatronic products, such as the automobile.

Millimeter wave radar technology has recently found applications in automobiles. The millimeter wave radar detects the location of objects (other vehicles) in the scenery and the distance to the obstacle and the velocity in real-time. A detailed description of a working system is given by Suzuki et al. [25]. Figure 1.4 shows an illustration of the vehicle-sensing capability with a millimeter-waver radar. This technology provides the capability to control the distance between the vehicle and an obstacle (or another vehicle) by integrating the sensor with the cruise control and ABS systems. The driver is able to set the speed and the desired distance between the cars ahead of him. The ABS system and the cruise control system are coupled together to safely achieve this remarkable capability. One logical extension of the obstacle avoidance capability is slow speed semi-autonomous driving where the vehicle maintains a constant distance from the vehicle ahead in traffic jam conditions. Fully autonomous vehicles are well within the scope of mechatronics development within the next 20 years. Supporting investigations are underway in many research centers on development of semi-autonomous cars with reactive path planning using GPS-based continuous traffic model updates and stop-and-go automation. A proposed sensing and control

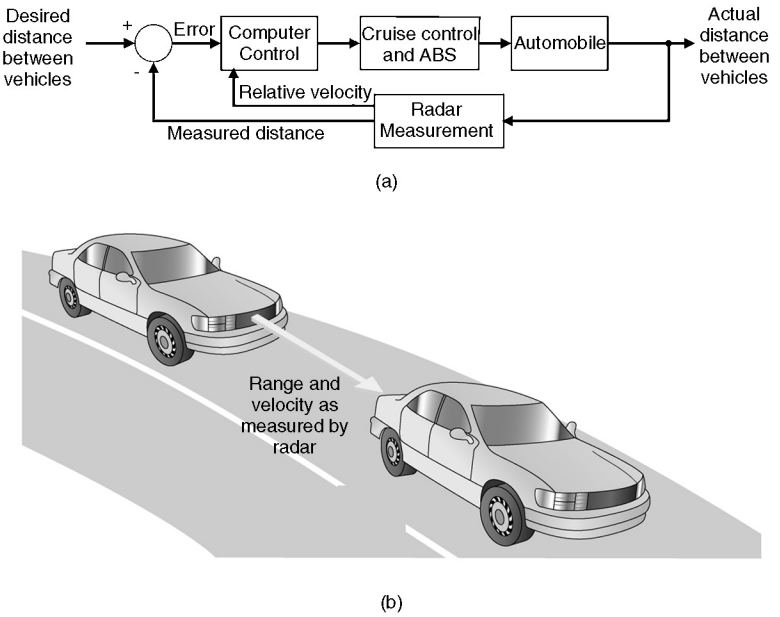



**FIGURE 1.4** Using a radar to measure distance and velocity to autonomously maintain desired distance between vehicles. (Adapted from *Modern Control Systems,* 9th ed., R. C. Dorf and R. H. Bishop, Prentice-Hall, 2001. Used with permission.)

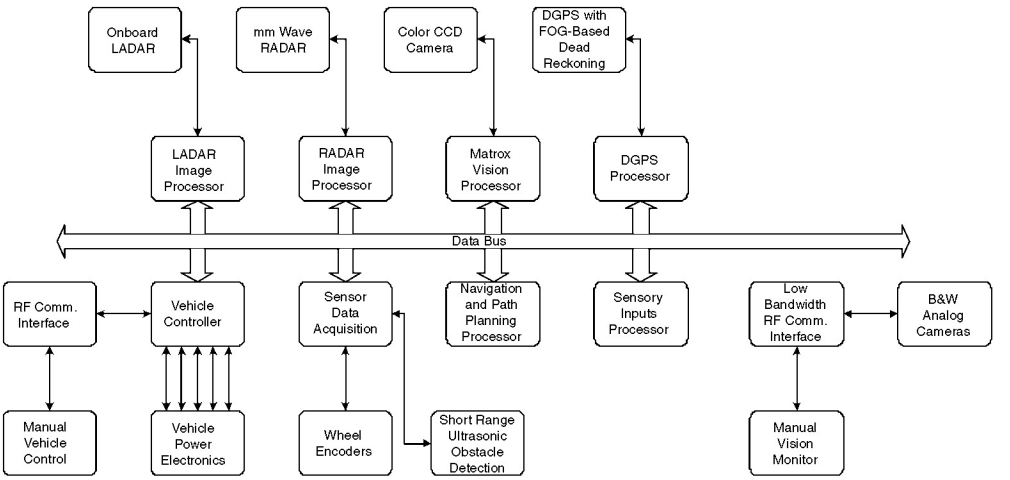



**FIGURE 1.5** Autonomous vehicle system design with sensors and actuators.

system for such a vehicle, shown in Figure 1.5, involves differential global positioning systems (DGPS), real-time image processing, and dynamic path planning [26].

Future mechatronic systems on automobiles may include a fog-free windshield based on humidity and temperature sensing and climate control, self-parallel parking, rear parking aid, lane change assistance, fluidless electronic brake-by-wire, and replacement of hydraulic systems with electromechanical servo systems. As the number of automobiles in the world increases, stricter emission standards are inevitable. Mechatronic products will in all likelihood contribute to meet the challenges in emission control and engine efficiency by providing substantial reduction in CO, NO, and HC emissions and increase in vehicle

efficiency [23]. Clearly, an automobile with 30–60 microcontrollers, up to 100 electric motors, about 200 pounds of wiring, a multitude of sensors, and thousands of lines of software code can hardly be classified as a strictly mechanical system. The automobile is being transformed into a comprehensive mechatronic system.

## 1.5 What is Mechatronics? And What's Next?

Mechatronics, the term coined in Japan in the 1970s, has evolved over the past 25 years and has led to a special breed of intelligent products. What is mechatronics? It is a natural stage in the evolutionary process of modern engineering design. For some engineers, mechatronics is nothing new, and, for others, it is a philosophical approach to design that serves as a guide for their activities. Certainly, mechatronics is an evolutionary process, not a revolutionary one. It is clear that an all-encompassing definition of mechatronics does not exist, but in reality, one is not needed. It is understood that mechatronics is about the synergistic integration of mechanical, electrical, and computer systems. One can understand the extent that mechatronics reaches into various disciplines by characterizing the constituent components comprising mechatronics, which include (i) physical systems modeling, (ii) sensors and actuators, (iii) signals and systems, (iv) computers and logic systems, and (v) software and data acquisition. Engineers and scientists from all walks of life and fields of study can contribute to mechatronics. As engineering and science boundaries become less well defined, more students will seek a multi-disciplinary education with a strong design component. Academia should be moving towards a curriculum, which includes coverage of mechatronic systems.

In the future, growth in mechatronic systems will be fueled by the growth in the constituent areas. Advancements in traditional disciplines fuel the growth of mechatronics systems by providing "enabling technologies." For example, the invention of the microprocessor had a profound effect on the redesign of mechanical systems and design of new mechatronics systems. We should expect continued advancements in cost-effective microprocessors and microcontrollers, sensor and actuator development enabled by advancements in applications of MEMS, adaptive control methodologies and real-time programming methods, networking and wireless technologies, mature CAE technologies for advanced system modeling, virtual prototyping, and testing. The continued rapid development in these areas will only accelerate the pace of smart product development. The Internet is a technology that, when utilized in combination with wireless technology, may also lead to new mechatronic products. While developments in automotives provide vivid examples of mechatronics development, there are numerous examples of intelligent systems in all walks of life, including smart home appliances such as dishwashers, vacuum cleaners, microwaves, and wireless network enabled devices. In the area of "human-friendly machines" (a term used by H. Kobayashi [27]), we can expect advances in robot-assisted surgery, and implantable sensors and actuators. Other areas that will benefit from mechatronic advances may include robotics, manufacturing, space technology, and transportation. The future of mechatronics is wide open.

### References

1. Kyura, N. and Oho, H., "Mechatronics—an industrial perspective," *IEEE/ASME Transactions on Mechatronics,* Vol. 1, No. 1, 1996, pp. 10–15.
2. Mori, T., "Mechatronics," Yasakawa Internal Trademark Application Memo 21.131.01, July 12, 1969.
3. Harshama, F., Tomizuka, M., and Fukuda, T., "Mechatronics—What is it, why, and how?—an editorial," *IEEE/ASME Transactions on Mechatronics,* Vol. 1, No. 1, 1996, pp. 1–4.
4. Auslander, D. M. and Kempf, C. J., *Mechatronics: Mechanical System Interfacing,* Prentice-Hall, Upper Saddle River, NJ, 1996.
5. Shetty, D. and Kolk, R. A., *Mechatronic System Design,* PWS Publishing Company, Boston, MA, 1997.
6. Bolton, W., *Mechatronics: Electrical Control Systems in Mechanical and Electrical Engineering, 2nd Ed.,* Addison-Wesley Longman, Harlow, England, 1999.
7. Mayr, I. O., *The Origins of Feedback Control,* MIT Press, Cambridge, MA, 1970.

8. Tomkinson, D. and Horne, J., *Mechatronics Engineering,* McGraw-Hill, New York, 1996.
9. Popov, E. P., *The Dynamics of Automatic Control Systems;* Gostekhizdat, Moscow, 1956; Addison-Wesley, Reading, MA, 1962.
10. Dorf, R. C. and Bishop, R. H., *Modern Control Systems, 9th Ed.,* Prentice-Hall, Upper Saddle River, NJ, 2000.
11. Maxwell, J. C., "On governors," *Proc. Royal Soc. London,* 16, 1868; in *Selected Papers on Mathematical Trends in Control Theory,* Dover, New York, 1964, pp. 270–283.
12. Vyshnegradskii, I. A., "On controllers of direct action," *Izv. SPB Tekhnotog. Inst.,* 1877.
13. Bode, H. W., "Feedback—the history of an idea," in *Selected Papers on Mathematical Trends in Control Theory,* Dover, New York, 1964, pp. 106–123.
14. Black, H. S., "Inventing the Negative Feedback Amplifier," *IEEE Spectrum,* December 1977, pp. 55–60.
15. Brittain, J. E., *Turning Points in American Electrical History,* IEEE Press, New York, 1977.
16. Fagen, M. D., *A History of Engineering and Science on the Bell Systems,* Bell Telephone Laboratories, 1978.
17. Newton, G., Gould, L., and Kaiser, J., *Analytical Design of Linear Feedback Control,* John Wiley & Sons, New York, 1957.
18. Dorf, R. C. and Kusiak, A., *Handbook of Automation and Manufacturing,* John Wiley & Sons, New York, 1994.
19. Dorf, R. C., *The Encyclopedia of Robotics,* John Wiley & Sons, New York, 1988.
20. Asami, K., Nomura, Y., and Naganawa, T., "Traction Control (TRC) System for 1987 Toyota Crown, 1989," *ABS-TCS-VDC Where Will the Technology Lead Us?* J. Mack, ed., Society of Automotive Engineers, Warrendale PA, 1996.
21. Pastor, S. et al., "Brake Control System," United States Patent # 5,720,533, Feb. 24, 1998 (see http://www.uspto.gov/ for more information).
22. Jorgensen, B., "Shifting gears," Auto Electronics, *Electronic Business,* Feb. 2001.
23. Barron, M. B. and Powers, W. F., "The role of electronic controls for future automotive mechatronic systems," *IEEE/ASME Transactions on Mechatronics,* Vol. 1, No. 1, 1996, pp. 80–88.
24. Kobe, G., "Electronics: What's driving the growth?" *Automotive Industries,* August 2000.
25. Suzuki, H., Hiroshi, M. Shono, and Isaji, O., "Radar Apparatus for Detecting a Distance/Velocity," United States Patent # 5,677,695, Oct 14, 1997 (see http://www.uspto.gov/ for more information).
26. Ramasubramanian, M. K., "Mechatronics—the future of mechanical engineering-past, present, and a vision for the future," (Invited paper), *Proc. SPIE,* Vol. 4334-34, March 2001.
27. Kobayashi, H. (Guest Editorial), *IEEE/ASME Transactions on Mechatronics,* Vol. 2, No. 4, 1997, p. 217.

# 2

# Mechatronic Design Approach




Rolf Isermann
*Darmstadt University of Technology*


## 2.1 Historical Development and Definition of Mechatronic Systems

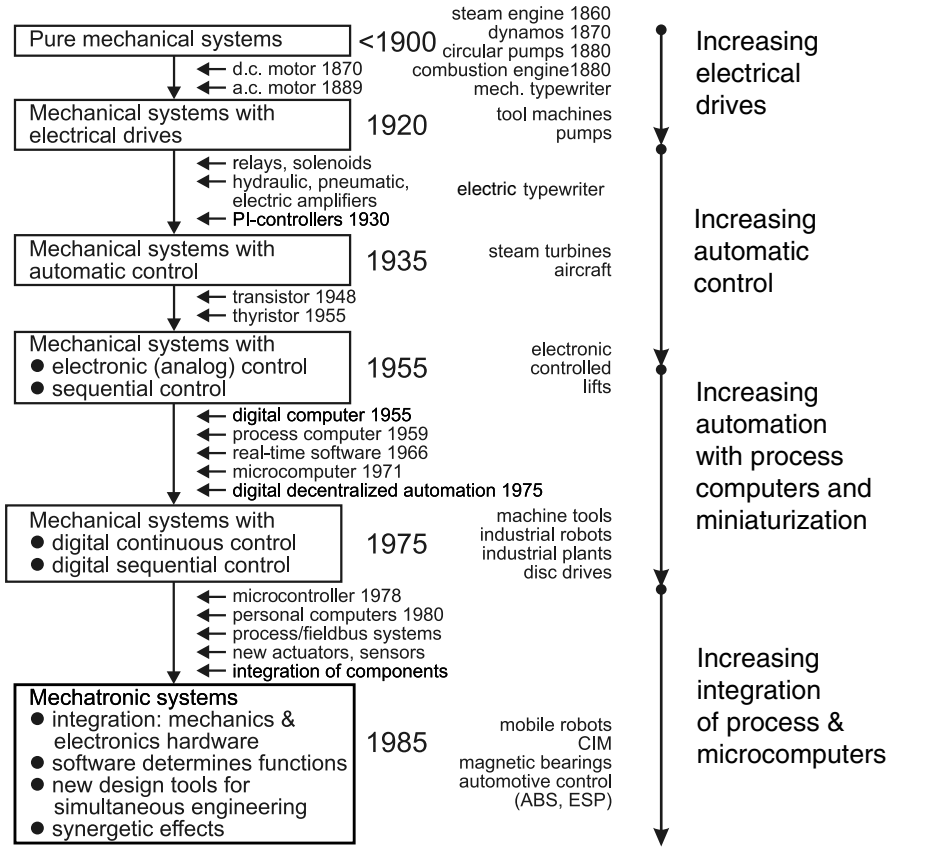
In several technical areas the integration of products or processes and electronics can be observed. This is especially true for mechanical systems which developed since about 1980. These systems changed from electro-mechanical systems with discrete electrical and mechanical parts to integrated electronic-mechanical systems with sensors, actuators, and digital microelectronics. These integrated systems, as seen in Table 2.1, are called *mechatronic systems*, with the connection of MECHAnics and elecTRONICS.

The word "mechatronics" was probably first created by a Japanese engineer in 1969 [1], with earlier definitions given by [2] and [3]. In [4], a preliminary definition is given: "Mechatronics is the synergetic integration of mechanical engineering with electronics and intelligent computer control in the design and manufacturing of industrial products and processes" [5].

All these definitions agree that mechatronics is an *interdisciplinary field*, in which the following disciplines act together (see Figure 2.1):

- *mechanical systems* (mechanical elements, machines, precision mechanics);
- *electronic systems* (microelectronics, power electronics, sensor and actuator technology); and
- *information technology* (systems theory, automation, software engineering, artificial intelligence).

**TABLE 2.1** Historical Development of Mechanical, Electrical, and Electronic Systems



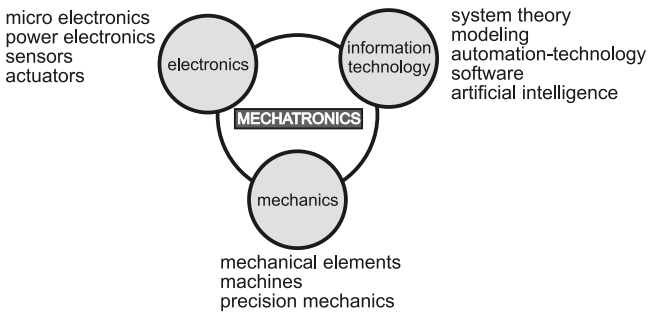




**FIGURE 2.1** Mechatronics: synergetic integration of different disciplines.

Some survey contributions describe the development of mechatronics; see [5–8]. An insight into general aspects are given in the journals [4,9,10]; first conference proceedings in [11–15]; and the books [16–19].

Figure 2.2 shows a general scheme of a modern mechanical process like a power producing or a power generating machine. A primary *energy flows* into the machine and is then either directly used for the energy consumer in the case of an energy transformer, or converted into another energy form in the case of an energy converter. The form of energy can be electrical, mechanical (potential or kinetic, hydraulic, pneumatic), chemical, or thermal. Machines are mostly characterized by a continuous or periodic (repetitive) energy flow. For other mechanical processes, such as mechanical elements or precision mechanical devices, piecewise or intermittent energy flows are typical.

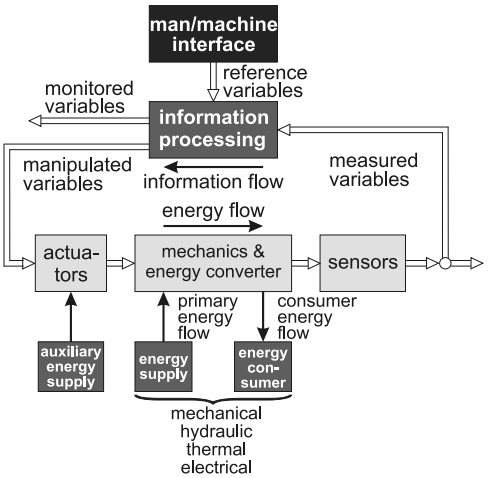



**FIGURE 2.2** Mechanical process and information processing develop towards mechatronic systems.

The energy flow is generally a product of a generalized flow and a potential (effort). Information on the state of the mechanical process can be obtained by measured generalized flows (speed, volume, or mass flow) or electrical current or potentials (force, pressure, temperature, or voltage). Together with reference variables, the measured variables are the inputs for an *information flow* through the digital electronics resulting in manipulated variables for the actuators or in monitored variables on a display.

The addition and integration of feedback information flow to a feedforward energy flow in a basically mechanical system is one characteristic of many mechatronic systems. This development presently influences the design of mechanical systems. Mechatronic systems can be subdivided into:

- mechatronic systems
- mechatronic machines
- mechatronic vehicles
- precision mechatronics
- micro mechatronics

This shows that the integration with electronics comprises many classes of technical systems. In several cases, the mechanical part of the process is coupled with an electrical, thermal, thermodynamic, chemical, or information processing part. This holds especially true for energy converters as machines where, in addition to the mechanical energy, other kinds of energy appear. Therefore, *mechatronic systems in a wider sense* comprise mechanical and also non-mechanical processes. However, the mechanical part normally dominates the system.

Because an auxiliary energy is required to change the fixed properties of formerly passive mechanical systems by feedforward or feedback control, these systems are sometimes also called *active mechanical systems*.

## 2.2 Functions of Mechatronic Systems

Mechatronic systems permit many improved and new functions. This will be discussed by considering some examples.

### Division of Functions between Mechanics and Electronics

For designing mechatronic systems, the interplay for the realization of functions in the mechanical and electronic part is crucial. Compared to pure mechanical realizations, the use of amplifiers and actuators with electrical auxiliary energy led to considerable simplifications in devices, as can be seen from watches,

electrical typewriters, and cameras. A further considerable *simplification in the mechanics* resulted from introducing microcomputers in connection with decentralized electrical drives, as can be seen from electronic typewriters, sewing machines, multi-axis handling systems, and automatic gears.

The design of lightweight constructions leads to elastic systems which are weakly damped through the material. An *electronic damping* through position, speed, or vibration sensors and electronic feedback can be realized with the additional advantage of an adjustable damping through the algorithms. Examples are elastic drive chains of vehicles with damping algorithms in the engine electronics, elastic robots, hydraulic systems, far reaching cranes, and space constructions (with, for example, flywheels).

The addition of closed loop control for position, speed, or force not only results in a precise tracking of reference variables, but also an approximate linear behavior, even though the mechanical systems show nonlinear behavior. By *omitting the constraint of linearization* on the mechanical side, the effort for construction and manufacturing may be reduced. Examples are simple mechanical pneumatic and electromechanical actuators and flow valves with electronic control.

With the aid of freely *programmable reference variable generation* the adaptation of nonlinear mechanical systems to the operator can be improved. This is already used for the driving pedal characteristics within the engine electronics for automobiles, telemanipulation of vehicles and aircraft, in development of hydraulic actuated excavators, and electric power steering.

With an increasing number of sensors, actuators, switches, and control units, the cable and electrical connections increase such that reliability, cost, weight, and the required space are major concerns. Therefore, the development of suitable bus systems, plug systems, and redundant and reconfigurable electronic systems are challenges for the designer.

## Improvement of Operating Properties

By applying active feedback control, precision is obtained not only through the high mechanical precision of a passively feedforward controlled mechanical element, but by comparison of a programmed reference variable and a measured control variable. Therefore, the mechanical precision in design and manufacturing may be reduced somewhat and more simple constructions for bearings or slideways can be used. An important aspect is the compensation of a larger and time variant friction by *adaptive friction compensation* [13,20]. Also, a larger friction on cost of backlash may be intended (such as gears with pretension), because it is usually easier to compensate for friction than for backlash.

*Model-based* and *adaptive control* allow for a wide range of operation, compared to fixed control with unsatisfactory performance (danger of instability or sluggish behavior). A combination of robust and adaptive control allows a wide range of operation for flow-, force-, or speed-control, and for processes like engines, vehicles, or aircraft. A better control performance allows the reference variables to move closer to the constraints with an improvement in efficiencies and yields (e.g., higher temperatures, pressures for combustion engines and turbines, compressors at stalling limits, higher tensions and higher speed for paper machines and steel mills).

## Addition of New Functions

Mechatronic systems allow functions to occur that could not be performed without digital electronics. First, *nonmeasurable quantities* can be calculated on the basis of measured signals and influenced by feedforward or feedback control. Examples are time-dependent variables such as slip for tyres, internal tensities, temperatures, slip angle and ground speed for steering control of vehicles, or parameters like damping, stiffness coefficients, and resistances. The *adaptation of parameters* such as damping and stiffness for oscillating systems (based on measurements of displacements or accelerations) is another example. Integrated *supervision and fault diagnosis* becomes more and more important with increasing automatic functions, increasing complexity, and higher demands on reliability and safety. Then, the triggering of redundant components, system reconfiguration, maintenance-on-request, and any kind of *teleservice* make the system more "intelligent." Table 2.2 summarizes some properties of mechatronic systems compared to conventional electro-mechanical systems.

**TABLE 2.2** Properties of Conventional and Mechatronic Design Systems

| Conventional Design | Mechatronic Design |
|---|---|
| **Added components** | **Integration of components (hardware)** |
| 1 Bulky | Compact |
| 2 Complex mechanisms | Simple mechanisms |
| 3 Cable problems | Bus or wireless communication |
| 4 Connected components | Autonomous units |
| **Simple control** | **Integration by information processing (software)** |
| 5 Stiff construction | Elastic construction with damping by electronic feedback |
| 6 Feedforward control, linear (analog) control | Programmable feedback (nonlinear) digital control |
| 7 Precision through narrow tolerances | Precision through measurement and feedback control |
| 8 Nonmeasurable quantities change arbitrarily | Control of nonmeasurable estimated quantities |
| 9 Simple monitoring | Supervision with fault diagnosis |
| 10 Fixed abilities | Learning abilities |

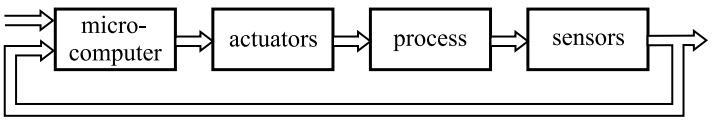



**FIGURE 2.3** General scheme of a (classical) mechanical-electronic system.

## 2.3 Ways of Integration

Figure 2.3 shows a general scheme of a classical mechanical-electronic system. Such systems resulted from adding available sensors, actuators, and analog or digital controllers to mechanical components. The limits of this approach were given by the lack of suitable sensors and actuators, the unsatisfactory life time under rough operating conditions (acceleration, temperature, contamination), the large space requirements, the required cables, and relatively slow data processing. With increasing improvements in miniaturization, robustness, and computing power of microelectronic components, one can now put more emphasis on electronics in the design of a mechatronic system. More autonomous systems can be envisioned, such as capsuled units with touchless signal transfer or bus connections, and robust microelectronics.

The integration within a mechatronic system can be performed through the integration of components and through the integration of information processing.

### Integration of Components (Hardware)

The integration of components (hardware integration) results from designing the mechatronic system as an overall system and imbedding the sensors, actuators, and microcomputers into the mechanical process, as seen in Figure 2.4. This spatial integration may be limited to the process and sensor, or to the process and actuator. Microcomputers can be integrated with the actuator, the process or sensor, or can be arranged at several places.

Integrated sensors and microcomputers lead to *smart sensors*, and integrated actuators and microcomputers lead to *smart actuators*. For larger systems, bus connections will replace cables. Hence, there are several possibilities to build up an integrated overall system by proper integration of the hardware.

### Integration of Information Processing (Software)

The integration of information processing (software integration) is mostly based on advanced control functions. Besides a basic feedforward and feedback control, an additional influence may take place through the process knowledge and corresponding online information processing, as seen in Figure 2.4. This means a processing of available signals at higher levels, including the solution of tasks like supervision

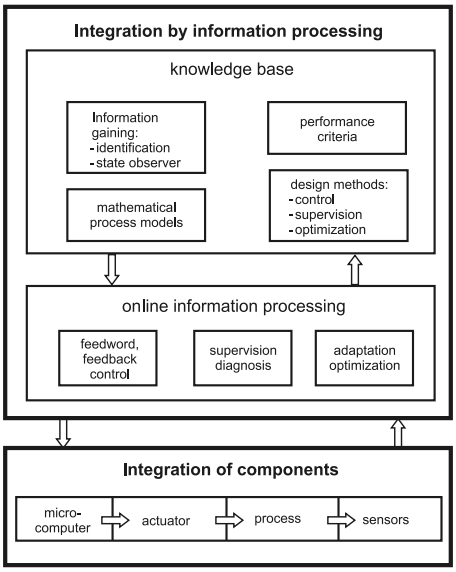



**FIGURE 2.4** Ways of integration within mechatronic systems.

with fault diagnosis, optimization, and general process management. The respective problem solutions result in real-time algorithms which must be adapted to the mechanical process properties, expressed by mathematical models in the form of static characteristics, or differential equations. Therefore, a *knowledge base* is required, comprising methods for design and information gaining, process models, and performance criteria. In this way, the mechanical parts are governed in various ways through higher level information processing with intelligent properties, possibly including learning, thus forming an integration by process-adapted software.

## 2.4 Information Processing Systems (Basic Architecture and HW/SW Trade-Offs)

The governing of mechanical systems is usually performed through actuators for the changing of positions, speeds, flows, forces, torques, and voltages. The directly measurable output quantities are frequently positions, speeds, accelerations, forces, and currents.

### Multilevel Control Architecture

The information processing of *direct measurable input and output signals* can be organized in several levels, as compared in Figure 2.5.

level 1: low level control (feedforward, feedback for damping, stabilization, linearization)
level 2: high level control (advanced feedback control strategies)
level 3: supervision, including fault diagnosis
level 4: optimization, coordination (of processes)
level 5: general process management

Recent approaches to mechatronic systems use signal processing in the lower levels, such as damping, control of motions, or simple supervision. Digital information processing, however, allows for the solution of many tasks, like adaptive control, learning control, supervision with fault diagnosis, decisions

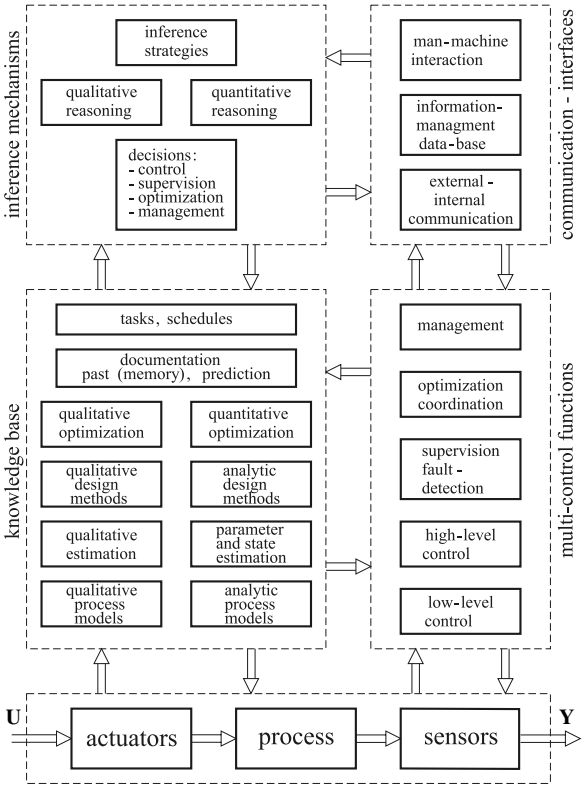



**FIGURE 2.5** Advanced intelligent automatic system with multi-control levels, knowledge base, inference mechanisms, and interfaces.

for maintenance or even redundancy actions, economic optimization, and coordination. The tasks of the higher levels are sometimes summarized as "process management."

## Special Signal Processing

The described methods are partially applicable for *nonmeasurable quantities* that are reconstructed from mathematical process models. In this way, it is possible to control damping ratios, material and heat stress, and slip, or to supervise quantities like resistances, capacitances, temperatures within components, or parameters of wear and contamination. This signal processing may require *special filters* to determine amplitudes or frequencies of vibrations, to determine derivated or integrated quantities, or *state variable observers*.

## Model-Based and Adaptive Control Systems

The information processing is, at least in the lower levels, performed by simple algorithms or software-modules under real-time conditions. These algorithms contain free adjustable parameters, which have to be adapted to the static and dynamic behavior of the process. In contrast to manual tuning by trial and error, the use of mathematical models allows precise and fast automatic adaptation.

The mathematical models can be obtained by identification and parameter estimation, which use the measured and sampled input and output signals. These methods are not restricted to linear models, but also allow for several classes of nonlinear systems. If the parameter estimation methods are combined with appropriate control algorithm design methods, adaptive control systems result. They can be used for permanent precise controller tuning or only for commissioning [20].

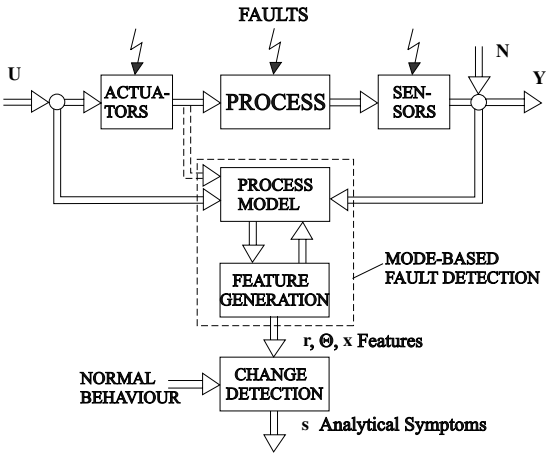



**FIGURE 2.6** Scheme for a model-based fault detection.

## Supervision and Fault Detection

With an increasing number of automatic functions (autonomy), including electronic components, sensors and actuators, increasing complexity, and increasing demands on reliability and safety, an integrated supervision with fault diagnosis becomes more and more important. This is a significant natural feature of an intelligent mechatronic system. Figure 2.6 shows a process influenced by faults. These faults indicate unpermitted deviations from normal states and can be generated either externally or internally. External faults can be caused by the power supply, contamination, or collision, internal faults by wear, missing lubrication, or actuator or sensor faults. The classical way for fault detection is the limit value checking of some few measurable variables. However, incipient and intermittant faults can not usually be detected, and an in-depth fault diagnosis is not possible by this simple approach. *Model-based fault detection* and *diagnosis methods* were developed in recent years, allowing for early detection of small faults with normally measured signals, also in closed loops [21]. Based on measured input signals, $\mathbf{U}(t)$, and output signals, $\mathbf{Y}(t)$, and process models, features are generated by parameter estimation, state and output observers, and parity equations, as seen in Figure 2.6.

These residuals are then compared with the residuals for normal behavior and with change detection methods analytical symptoms are obtained. Then, a fault diagnosis is performed via methods of classification or reasoning. For further details see [22,23].

A considerable advantage is if the same process model can be used for both the (adaptive) *controller design and the fault detection*. In general, continuous time models are preferred if fault detection is based on parameter estimation or parity equations. For fault detection with state estimation or parity equations, discrete-time models can be used.

Advanced supervision and fault diagnosis is a basis for improving reliability and safety, state dependent maintenance, triggering of redundancies, and reconfiguration.

## Intelligent Systems (Basic Tasks)

The information processing within mechatronic systems may range between simple control functions and intelligent control. Various definitions of intelligent control systems do exist, see [24–30]. An intelligent control system may be organized as an *online expert system*, according to Figure 2.5, and comprises

- multi-control functions (executive functions),
- a knowledge base,
- inference mechanisms, and
- communication interfaces.

The online *control functions* are usually organized in multilevels, as already described. The *knowledge base* contains quantitative and qualitative knowledge. The quantitative part operates with analytic (mathematical) process models, parameter and state estimation methods, analytic design methods (e.g., for control and fault detection), and quantitative optimization methods. Similar modules hold for the qualitative knowledge (e.g., in the form of rules for fuzzy and soft computing). Further knowledge is the past history in the memory and the possibility to predict the behavior. Finally, tasks or schedules may be included.

The *inference mechanism* draws conclusions either by quantitative reasoning (e.g., Boolean methods) or by qualitative reasoning (e.g., possibilistic methods) and takes decisions for the executive functions.

Communication between the different modules, an information management database, and the man–machine interaction has to be organized.

Based on these functions of an online expert system, an intelligent system can be built up, with the ability "to model, reason and learn the process and its automatic functions within a given frame and to govern it towards a certain goal." Hence, intelligent mechatronic systems can be developed, ranging from "low-degree intelligent" [13], such as intelligent actuators, to "fairly intelligent systems," such as self-navigating automatic guided vehicles.

An *intelligent mechatronic system* adapts the controller to the mostly nonlinear behavior (adaptation), and stores its controller parameters in dependence on the position and load (learning), supervises all relevant elements, and performs a fault diagnosis (supervision) to request maintenance or, if a failure occurs, to request a fail safe action (decisions on actions). In the case of multiple components, supervision may help to switch off the faulty component and to perform a reconfiguration of the controlled process.

## 2.5 Concurrent Design Procedure for Mechatronic Systems

The design of mechatronic systems requires a systematic development and use of modern design tools.

### Design Steps

Table 2.3 shows five important development steps for mechatronic systems, starting from a purely mechanical system and resulting in a fully integrated mechatronic system. Depending on the kind of mechanical system, the intensity of the single development steps is different. For precision mechanical devices, fairly integrated mechatronic systems do exist. The influence of the electronics on *mechanical elements* may be considerable, as shown by adaptive dampers, anti-lock system brakes, and automatic gears. However, complete *machines* and *vehicles* show first a mechatronic design of their elements, and then slowly a redesign of parts of the overall structure as can be observed in the development of machine tools, robots, and vehicle bodies.

### Required CAD/CAE Tools

The computer aided development of mechatronic systems comprises:

1. constructive specification in the engineering development stage using CAD and CAE tools,
2. model building for obtaining static and dynamic process models,
3. transformation into computer codes for system simulation, and
4. programming and implementation of the final mechatronic software.

Some software tools are described in [31]. A broad range of CAD/CAE tools is available for 2D- and 3D-mechanical design, such as Auto CAD with a direct link to CAM (computer-aided manufacturing), and PADS, for multilayer, printed-circuit board layout. However, the state of computer-aided modeling is not as advanced. Object-oriented languages such as DYMOLA and MOBILE for modeling of large combined systems are described in [31–33]. These packages are based on specified ordinary differential

**TABLE 2.3** Steps in the Design of Mechatronic Systems

| | Precision Mechanics | Mechanical Elements | Machines |
|---|---|---|---|
| Pure mechanical system |  |  |  |
| 1. Addition of sensors, actuators, microelectronics, control functions |  |  |  |
| 2. Integration of components (hardware integration) |  |  |  |
| 3. Integration by information processing (software integration) |  |  |  |
| 4. Redesign of mechanical system |  |  |  |
| 5. Creation of synergetic effects |  |  |  |
| Fully integrated mechatronic systems |  |  |  |
| Examples | Sensors actuators disc-storages cameras | Suspensions dampers clutches gears brakes | Electric drives combustion engines mach. tools robots |

The size of a circle indicates the present intensity of the respective mechatronic development step:  large,  medium,  little.

equations, algebraic equations, and discontinuities. A recent description of the state of computer-aided control system design can be found in [34]. For system simulation (and controller design), a variety of program systems exist, like ACSL, SIMPACK, MATLAB/SIMULINK, and MATRIX-X. These simulation techniques are valuable tools for design, as they allow the designer to study the interaction of components and the variations of design parameters before manufacturing. They are, in general, not suitable for real-time simulation.

## Modeling Procedure

Mathematical process models for static and dynamic behavior are required for various steps in the design of mechatronic systems, such as simulation, control design, and reconstruction of variables. Two ways to obtain these models are *theoretical modeling* based on first (physical) principles and *experimental modeling* (*identification*) with measured input and output variables. A basic problem of theoretical modeling of mechatronic systems is that the components originate from different domains. There exists a well-developed domain specific knowledge for the modeling of electrical circuits, multibody mechanical systems, or hydraulic systems, and corresponding software packages. However, a computer-assisted general methodology for the modeling and simulation of components from different domains is still missing [35].

The basic principles of theoretical modeling for system with energy flow are known and can be unified for components from different domains as electrical, mechanical, and thermal (see [36–41]). The modeling methodology becomes more involved if material flows are incorporated as for fluidics, thermodynamics, and chemical processes.

A general procedure for theoretical modeling of lumped parameter processes can be sketched as follows [19].

1. Definition of flows
   - energy flow (electrical, mechanical, thermal conductance)
   - energy and material flow (fluidic, thermal transfer, thermodynamic, chemical)
2. Definition of process elements: flow diagrams
   - sources, sinks (dissipative)
   - storages, transformers, converters
3. Graphical representation of the process model
   - multi-port diagrams (terminals, flows, and potentials, or across and through variables)
   - block diagrams for signal flow
   - bond graphs for energy flow
4. Statement of equations for all process elements
   (i) Balance equations for storage (mass, energy, momentum)
   (ii) Constitutive equations for process elements (sources, transformers, converters)
   (iii) Phenomenological laws for irreversible processes (dissipative systems: sinks)
5. Interconnection equations for the process elements
   - continuity equations for parallel connections (node law)
   - compatibility equations for serial connections (closed circuit law)
6. Overall process model calculation
   - establishment of input and output variables
   - state space representation
   - input/output models (differential equations, transfer functions)

An example of steps 1–3 is shown in Figure 2.7 for a drive-by-wire vehicle. A unified approach for processes with energy flow is known for electrical, mechanical, and hydraulic processes with incompressible fluids. Table 2.4 defines generalized through and across variables.

In these cases, the product of the through and across variable is power. This unification enabled the formulation of the standard *bond graph modeling* [39]. Also, for hydraulic processes with compressible fluids and thermal processes, these variables can be defined to result in powers, as seen in Table 2.4. However, using mass flows and heat flows is not engineering practice. If these variables are used, so-called pseudo bond graphs with special laws result, leaving the simplicity of standard bond graphs. Bond graphs lead to a high-level abstraction, have less flexibility, and need additional effort to generate simulation algorithms. Therefore, they are not the ideal tool for mechatronic systems [35]. Also, the tedious work needed to establish *block diagrams* with an early definition of causal input/output blocks is not suitable.

Development towards object-oriented modeling is on the way, where objects with terminals (cuts) are defined without assuming a causality in this basic state. Then, object diagrams are graphically represented, retaining an intuitive understanding of the original physical components [43,44]. Hence, theoretical modeling of mechatronic systems with a unified, transparent, and flexible procedure (from the basic components of different domains to simulation) are a challenge for further development. Many components show nonlinear behavior and nonlinearities (friction and backlash). For more complex process parts, multidimensional mappings (e.g., combustion engines, tire behavior) must be integrated.

For verification of theoretical models, several well-known identification methods can be used, such as correlation analysis and frequency response measurement, or Fourier- and spectral analysis. Since some parameters are unknown or changed with time, parameter estimation methods can be applied, both, for models with continuous time or discrete time (especially if the models are linear in the parameters) [42,45,46]. For the identification and approximation of nonlinear, multi-dimensional characteristics,

**TABLE 2.4** Generalized Through and Across Variables for Processes with Energy Flow

| System | Through Variables | | Across Variables | |
|---|---|---|---|---|
| Electrical | Electric current | $I$ | Electric voltage | $U$ |
| Magnetic | Magnetic Flow | $\Phi$ | Magnetic force | $\Theta$ |
| Mechanical | | | | |
| • translation | Force | $F$ | Velocity | $w$ |
| • rotation | Torque | $M$ | Rotational speed | $\omega$ |
| Hydraulic | Volume flow | $\dot{V}$ | Pressure | $p$ |
| Thermodynamic | Entropy flow | | Temperature | $T$ |

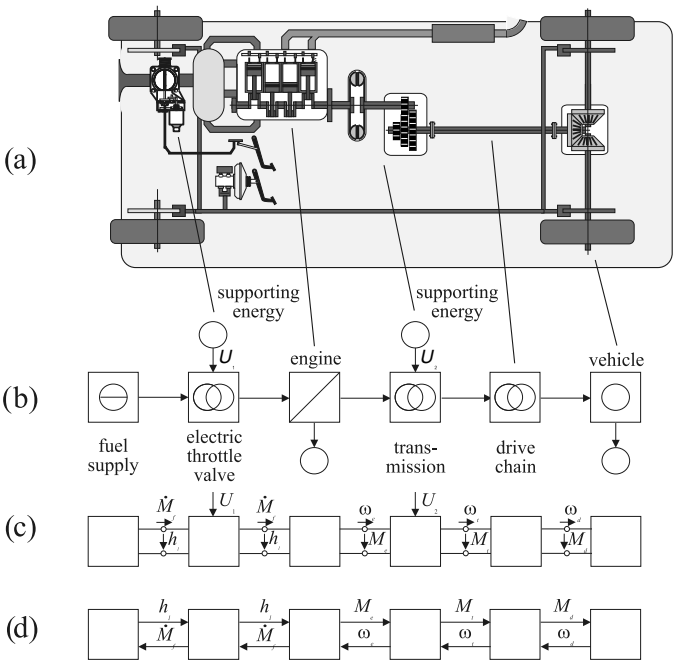



**FIGURE 2.7** Different schemes for an automobile (as required for drive-by-wire-longitudinal control): (a) scheme of the components (construction map), (b) energy flow diagram (simplified), (c) multi-port diagram with flows and potentials, (d) signal flow diagram for multi-ports.

artificial neural networks (multilayer perceptrons or radial-basis-functions) can be expanded for non-linear dynamic processes [47].

## Real-Time Simulation

Increasingly, real-time simulation is applied to the design of mechatronic systems. This is especially true if the process, the hardware, and the software are developed simultaneously in order to minimize iterative development cycles and to meet short time-to-market schedules. With regard to the required speed of computation *simulation methods*, it can be subdivided into

1. simulation without (hard) time limitation,
2. real-time simulation, and
3. simulation faster than real-time.

Some application examples are given in Figure 2.8. Herewith, *real-time simulation* means that the simulation of a component is performed such that the input and output signals show the same

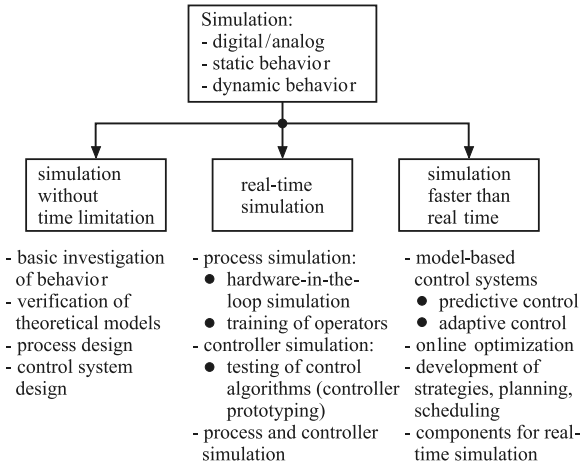



**FIGURE 2.8** Classification of simulation methods with regard to speed and application examples.

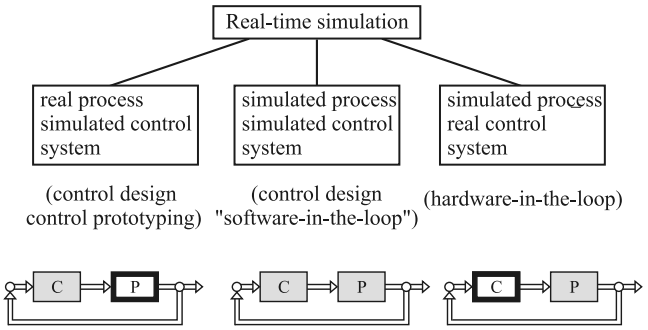



**FIGURE 2.9** Classification of real-time simulation.

time-dependent values as the real, dynamically operating component. This becomes a computational problem for processes which have fast dynamics compared to the required algorithms and calculation speed.

Different kinds of real-time simulation methods are shown in Figure 2.9. The reason for the real-time requirement is mostly that one part of the investigated system is not simulated but real. Three cases can be distinguished:

1. The *real process* can be operated together with the *simulated control* by using hardware other than the final hardware. This is also called "control prototyping."
2. The *simulated process* can be operated with the *real control hardware*, which is called "hardware-in-the-loop simulation."
3. The *simulated process* is run with the *simulated control* in real time. This may be required if the final hardware is not available or if a design step before the hardware-in-the-loop simulation is considered.

## Hardware-in-the-Loop Simulation

The *hardware-in-the-loop* simulation (HIL) is characterized by operating real components in connection with real-time simulated components. Usually, the control system hardware and software is the real system, as used for series production. The controlled process (consisting of actuators, physical processes, and sensors) can either comprise simulated components or real components, as seen in Figure 2.10(a). In general, mixtures of the shown cases are realized. Frequently, some actuators are real and the process

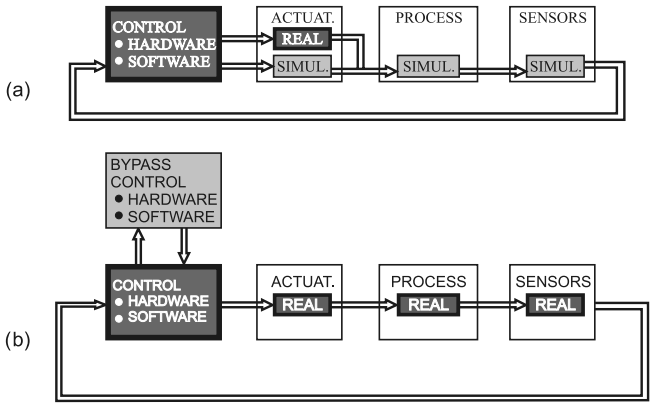



**FIGURE 2.10** Real-time simulation: hybrid structures. (a) Hardware-in-the-loop simulation. (b) Control prototyping.

and the sensors are simulated. The reason is that actuators and the control hardware very often form one integrated subsystem or that actuators are difficult to model precisely and to simulate in real time. (The use of real sensors together with a simulated process may require considerable realization efforts, because the physical sensor input does not exist and must be generated artificially.) In order to change or redesign some functions of the control hardware or software, a bypass unit can be connected to the basic control hardware. Hence, hardware-in-the-loop simulators may also contain partially simulated (emulated) control functions.

The advantages of the hardware-in-the-loop simulation are generally:

- design and testing of the control hardware and software without operating a real process ("moving the process field into the laboratory");
- testing of the control hardware and software under extreme environmental conditions in the laboratory (e.g., high/low temperature, high accelerations and mechanical shocks, aggressive media, electro-magnetic compatibility);
- testing of the effects of faults and failures of actuators, sensors, and computers on the overall system;
- operating and testing of extreme and dangerous operating conditions;
- reproducible experiments, frequently repeatable;
- easy operation with different man-machine interfaces (cockpit-design and training of operators); and
- saving of cost and development time.

## Control Prototyping

For the design and testing of complex control systems and their algorithms under real-time constraints, a real-time controller simulation (emulation) with hardware (e.g., off-the-shelf signal processor) other than the final series production hardware (e.g., special ASICS) may be performed. The process, the actuators, and sensors can then be real. This is called *control prototyping* (Figure 2.10(b)). However, parts of the process or actuators may be simulated, resulting in a mixture of HIL-simulation and control prototyping. The advantages are mainly:

- early development of signal processing methods, process models, and control system structure, including algorithms with high level software and high performance off-the-shelf hardware;
- testing of signal processing and control systems, together with other design of actuators, process parts, and sensor technology, in order to create synergetic effects;

- reduction of models and algorithms to meet the requirements of cheaper mass production hardware; and
- defining the specifications for final hardware and software.

Some of the advantages of HIL-simulation also hold for control prototyping. Some references for real-time simulation are [48,49].

## References


1. Kyura, N. and Oho, H., Mechatronics—an industrial perspective. *IEEE/ASME Transactions on Mechatronics,* 1(1):10–15.
2. Schweitzer, G., Mechatronik-Aufgaben und Lösungen. VDI-Berichte Nr. 787. VDI-Verlag, Düsseldorf, 1989.
3. Ovaska, S. J., Electronics and information technology in high range elevator systems. *Mechatronics,* 2(1):89–99, 1992.
4. *IEEE/ASME Transactions on Mechatronics,* 1996.
5. Harashima, F., Tomizuka, M., and Fukuda, T., Mechatronics—"What is it, why and how?" An editorial. *IEEE/ASME Transactions on Mechatronics,* 1(1):1–4, 1996.
6. Schweitzer, G., Mechatronics—a concept with examples in active magnetic bearings. *Mechatronics,* 2(1):65–74, 1992.
7. Gausemeier, J., Brexel, D., Frank, Th., and Humpert, A., Integrated product development. In *Third Conf. Mechatronics and Robotics,* Paderborn, Germany, Okt. 4–6, 1995. Teubner, Stuttgart, 1995.
8. Isermann, R., Modeling and design methodology for mechatronic systems. *IEEE/ASME Transactions on Mechatronics,* 1(1):16–28, 1996.
9. *Mechatronics: An International Journal. Aims and Scope.* Pergamon Press, Oxford, 1991.
10. *Mechatronics Systems Engineering: International Journal on Design and Application of Integrated Electromechanical Systems.* Kluwer Academic Publishers, Nethol, 1993.
11. IEE, Mechatronics: Designing intelligent machines. In *Proc. IEE-Int. Conf.* 12–13 Sep., Univ. of Cambridge, 1990.
12. Hiller, M. (ed.), *Second Conf. Mechatronics and Robotics.* September 27–29, Duisburg/Moers, Germany, 1993. Moers, IMECH, 1993.
13. Isermann, R. (ed.), Integrierte mechanisch elektroni-sche Systeme. March 2–3, Darmstadt, Germany, 1993. Fortschr.-Ber. VDI Reihe 12 Nr. 179. VDI-Verlag, Düsseldorf, 1993.
14. Lückel, J. (ed.), *Third Conf. Mechatronics and Robotics,* Paderborn, Germany, Oct. 4–6, 1995. Teubner, Stuttgart, 1995.
15. Kaynak, O., Özkan, M., Bekiroglu, N., and Tunay, I. (eds.), Recent advances in mechatronics. In *Proc. Int. Conf. Recent Advances in Mechatronics,* August 14–16, 1995, Istanbul, Turkey.
16. Kitaura, K., Industrial mechatronics. New East Business Ltd., in Japanese, 1991.
17. Bradley, D. A., Dawson, D., Burd, D., and Loader, A. J., *Mechatronics-Electronics in Products and Processes.* Chapman and Hall, London, 1991.
18. McConaill, P. A., Drews, P., and Robrock, K. H., *Mechatronics and Robotics I.* IOS-Press, Amsterdam, 1991.
19. Isermann, R., *Mechatronische Systeme.* Springer, Berlin, 1999.
20. Isermann, R., Lachmann, K. H., and Matko, D., *Adaptive Control Systems,* Prentice-Hall, London, 1992.
21. Isermann, R., Supervision, fault detection and fault diagnosis methods—advanced methods and applications. In *Proc. XIV IMEKO World Congress,* Vol. 1, pp. 1–28, Tampere, Finland, 1997.
22. Isermann, R., Supervision, fault detection and fault diagnosis methods—an introduction, special section on supervision, fault detection and diagnosis. *Control Engineering Practice,* 5(5):639–652, 1997.
23. Isermann, R. (ed.), Special section on supervision, fault detection and diagnosis. *Control Engineering Practice,* 5(5):1997.

24. Saridis, G. N., *Self Organizing Control of Stochastic Systems.* Marcel Dekker, New York, 1977.
25. Saridis, G. N. and Valavanis, K. P., Analytical design of intelligent machines. *Automatica,* 24:123–133, 1988.
26. Åström, K. J., Intelligent control. In *Proc. European Control Conf.,* Grenoble, 1991.
27. White, D. A. and Sofge, D. A. (eds.), *Handbook of Intelligent Control.* Van Norstrad, Reinhold, New York, 1992.
28. Antaklis, P., Defining intelligent control. *IEEE Control Systems,* Vol. June: 4–66, 1994.
29. Gupta, M. M. and Sinha, N. K., *Intelligent Control Systems.* IEEE-Press, New York, 1996.
30. Harris, C. J. (ed.), *Advances in Intelligent Control.* Taylor & Francis, London, 1994.
31. Otter, M. and Gruebel, G., Direct physical modeling and automatic code generation for mechatronics simulation. In *Proc. 2nd Conf. Mechatronics and Robotics,* Duisburg, Sep. 27–29, IMECH, Moers, 1993.
32. Elmquist, H., Object-oriented modeling and automatic formula manipulation in Dymola, Scandin. Simul. Society SIMS, June, Kongsberg, 1993.
33. Hiller, M., Modelling, simulation and control design for large and heavy manipulators. In *Proc. Int. Conf. Recent Advances in Mechatronics.* 1:78–85, Istanbul, Turkey, 1995.
34. James, J., Cellier, F., Pang, G., Gray, J., and Mattson, S. E., The state of computer-aided control system design (CACSD). *IEEE Transactions on Control Systems,* Special Issue, April 6–7 (1995).
35. Otter, M. and Elmqvist, H., Energy flow modeling of mechatronic systems via object diagrams. In *Proc. 2nd MATHMOD,* Vienna, 705–710, 1997.
36. Paynter, H. M., *Analysis and Design of Engineering Systems.* MIT Press, Cambridge, 1961.
37. MacFarlane, A. G. J., *Engineering Systems Analysis.* G. G. Harrop, Cambridge, 1964.
38. Wellstead, P. E., *Introduction to Physical System Modelling.* Academic Press, London, 1979.
39. Karnopp, D. C., Margolis, D. L., and Rosenberg, R. C., *System Dynamics. A Unified Approach.* J. Wiley, New York, 1990.
40. Cellier, F. E., *Continuous System Modelling.* Springer, Berlin, 1991.
41. Gawtrop, F. E. and Smith, L., *Metamodelling: Bond Graphs and Dynamic Systems.* Prentice-Hall, London, 1996.
42. Eykhoff, P., *System Identification.* John Wiley & Sons, London, 1974.
43. Elmqvist, H., A structured model language for large continuous systems. Ph.D. Dissertation, Report CODEN: LUTFD2/(TFRT-1015) Dept. of Aut. Control, Lund Institute of Technology, Sweden, 1978.
44. Elmqvist, H. and Mattson, S. E., Simulator for dynamical systems using graphics and equations for modeling. *IEEE Control Systems Magazine,* 9(1):53–58, 1989.
45. Isermann, R., *Identifikation dynamischer Systeme.* 2nd Ed., Vol. 1 and 2. Springer, Berlin, 1992.
46. Ljung, L., *System Identification: Theory for the User.* Prentice-Hall, Englewood Cliffs, NJ, 1987.
47. Isermann, R., Ernst, S., and Nelles, O., Identification with dynamic neural networks—architectures, comparisons, applications—Plenary. In *Proc. IFAC Symp. System Identification (SYSID'97),* Vol. 3, pp. 997–1022, Fukuoka, Japan, 1997.
48. Hanselmann, H., Hardware-in-the-loop simulation as a standard approach for development, customization, and production test, SAE 930207, 1993.
49. Isermann, R., Schaffnit, J., and Sinsel, S., Hardware-in-the-loop simulation for the design and testing of engine control systems. *Control Engineering Practice,* 7(7):643–653, 1999.

# 3

# System Interfacing, Instrumentation, and Control Systems




Rick Homkes
*Purdue University*


## 3.1 Introduction

The purpose of this chapter is to introduce a number of topics dealing with a mechatronic system. This starts with an overview of mechatronic systems and a look at the input and output signals of a mechatronic system. The special features of microprocessor input and output are next. Software, an often-neglected portion of a mechatronic system, is briefly covered with an emphasis on software engineering concepts. The chapter concludes with a short discussion of testing and instrumentation.

## The Mechatronic System

Figure 3.1 shows a typical mechatronic system with mechanical, electrical, and computer components. The process of system data acquisition begins with the measurement of a physical value by a sensor. The sensor is able to generate some form of signal, generally an analog signal in the form of a voltage level or waveform. This analog signal is sent to an analog-to-digital converter (ADC). Commonly using a process of successive approximation, the ADC maps the analog input signal to a digital output. This digital value is composed of a set of binary values called bits (often represented by 0s and 1s). The set of bits represents a decimal or hexadecimal number that can be used by the microcontroller. The microcontroller consists of a microprocessor plus memory and other attached devices. The program in the microprocessor uses this digital value along with other inputs and preloaded values called calibrations to determine output commands. Like the input to the microprocessor, these outputs are in digital form and can be represented by a set of bits. A digital-to-analog converter (DAC) is then often used to convert the digital value into an analog signal. The analog signal is used by an actuator to control a physical device or affect the physical environment. The sensor then takes new measurements and the process repeated, thus completing a feedback control loop. Timing for this entire operation is synchronized by the use of a clock.

## A Home/Office Example

An example of a mechatronic system is the common heating/cooling system for homes and offices. Simple systems use a bimetal thermostat with contact points controlling a mercury switch that turns on and off the furnace or air conditioner. A modern environmental control system uses these same basic components along with other components and computer program control. A temperature sensor monitors the physical environment and produces a voltage level as demonstrated in Figure 3.2 (though generally not nearly such a smooth function). After conversion by the ADC, the microcontroller uses the digitized temperature

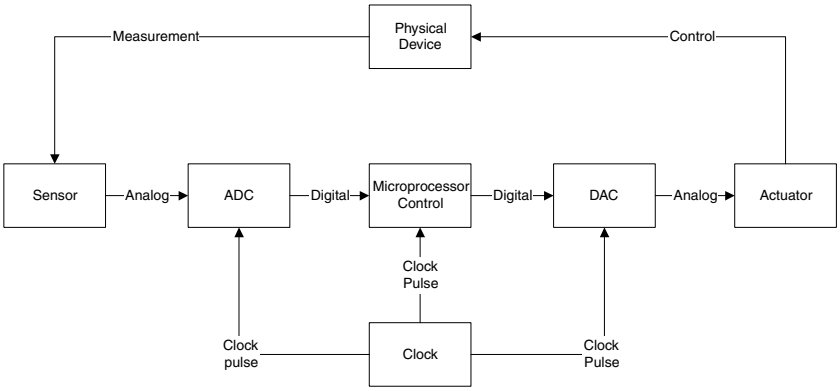



**FIGURE 3.1** Microprocessor control system.

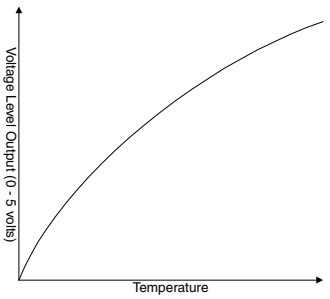



**FIGURE 3.2** Voltage levels.

data along with a 24-hour clock and the user requested temperatures to produce a digital control signal. This signal directs the actuator, usually a simple electrical switch in this example. The switch, in turn, controls a motor to turn the heating or cooling unit on or off. New measurements are then taken and the cycle is repeated. While not a mechatronic product on the order of a camcorder, it is a mechatronic system because of its combination of mechanical, electrical, and computer components. This system may also incorporate some additional features. If the temperature being sensed is quite high, say 80°C, it is possible that a fire exists. It is then not a good idea to turn on the blower fan and feed the fire more oxygen. Instead the system should set off an alarm or use a data communication device to alert the fire department. Because of this type of computer control, the system is "smart," at least relative to the older mercury-switch controlled systems.

### An Automotive Example

A second example is the Antilock Braking System (ABS) found in many vehicles. The entire purpose of this type of system is to prevent a wheel from locking up and thus having the driver loose directional control of the vehicle due to skidding. In this case, sensors attached to each wheel determine the rotational speed of the wheels. These data, probably in a waveform or time-varied electrical voltage, is sent to the microcontroller along with the data from sensors reporting inputs such as brake pedal position, vehicle speed, and yaw. After conversion by the ADC or input capture routine into a digital value, the program in the microprocessor then determines the necessary action. This is where the aspect of human computer interface (HCI) or human machine interface (HMI) comes into play by taking account of the "feel" of the system to the user. System calibration can adjust the response to the driver while, of course, stopping the vehicle by controlling the brakes with the actuators. There are two important things to note in this example. The first is that, in the end, the vehicle is being stopped because of hydraulic forces pressing the brake pad against a drum or rotor—a purely mechanical function. The other is that the ABS, while an "intelligent product," is not a stand-alone device. It is part of a larger system, the vehicle, with multiple microcontrollers working together through the data network of the vehicle.

## 3.2 Input Signals of a Mechatronic System

### Transducer/Sensor Input

All inputs to mechatronic systems come from either some form of sensory apparatus or communications from other systems. Sensors were first introduced in the previous section and will be discussed in much more depth in Chapter 19. Transducers, devices that convert energy from one form to another, are often used synonymously with sensors. Transducers and their properties will be explained fully in Chapter 45. Sensors can be divided into two general classifications, active or passive. Active sensors emit a signal in order to estimate an attribute of the environment or device being measured. Passive sensors do not. A military example of this difference would be a strike aircraft "painting" a target using either active laser radar (LADAR) or a passive forward looking infrared (FLIR) sensor.

As stated in the Introduction section, the output of a sensor is usually an analog signal. The simplest type of analog signal is a voltage level with a direct (though not necessarily linear) correlation to the input condition. A second type is a pulse width modulated (PWM) signal, which will be explained further in a later section of this chapter when discussing microcontroller outputs. A third type is a waveform, as shown in Figure 3.3. This type of signal is modulated either in its amplitude (Figure 3.4) or its frequency (Figure 3.5) or, in some cases, both. These changes reflect the changes in the condition being monitored.

There are sensors that do not produce an analog signal. Some of these sensors produce a square wave as in Figure 3.6 that is input to the microcontroller using the EIA 232 communications standard. The square wave represents the binary values of 0 and 1. In this case the ADC is probably on-board the sensor itself, adding to the cost of the sensor. Some sensors/recorders can even create mail or TCP/IP packets

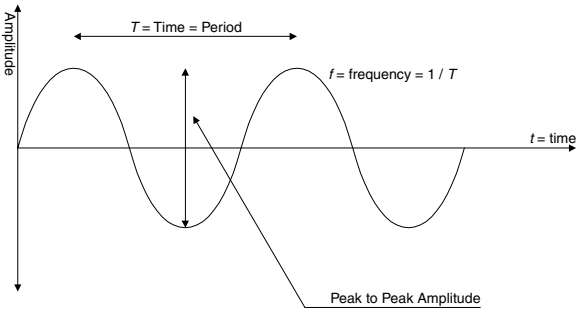



**FIGURE 3.3** Sine wave.

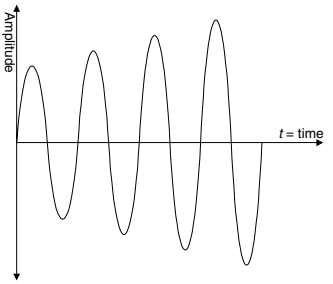



**FIGURE 3.4** Amplitude modulation.

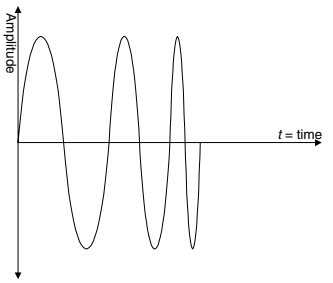



**FIGURE 3.5** Frequency modulation.

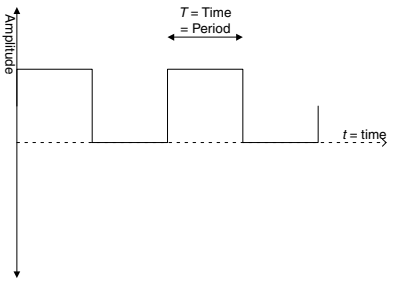



**FIGURE 3.6** Square wave.

as output. An example of this type of unit is the MV100 MobileCorder from Yokogawa Corporation of America.

### Analog-to-Digital Converters

The ADC can basically be typed by two parameters: the analog input range and the digital output range. As an example, consider an ADC that is converting a voltage level ranging 0–12 V into a single byte of 8 bits. In this example, each binary count increment reflects an increase in analog voltage of 1/256 of the maximum 12 V. There is an unusual twist to this conversion, however. Since a zero value represents 0 V, and a 128 value represents half of the maximum value, 6 V in this example, the maximum decimal value of 255 represents 255/256 of the maximum voltage value, or 11.953125 V. A table of the equivalent values is shown below:

| Binary | Decimal | Voltage |
|---|---|---|
| 0000 0000 | 0 | 0.0 |
| 0000 0001 | 1 | 0.00390625 |
| 1000 0000 | 128 | 6.0 |
| 1111 1111 | 255 | 11.953125 |

An ADC that is implemented in the Motorola HC12 microcontroller produces 10 bits. While not fitting so nicely into a single byte of data, this 10-bit ADC does give additional resolution. Using an input range from 0 to 5 V, the decimal resolution per least significant bit is 4.88 mV. If the ADC had 8 bits of output, the resolution per bit would be 19.5 mV, a fourfold difference. Larger voltages, e.g., from 0 to 12 V, can be scaled with a voltage divider to fit the 0–5 V range. Smaller voltages can be amplified to span the entire range. A process known as successive approximation (using the Successive Approximation Register or SAR in the Motorola chip) is used to determine the correct digital value.

## 3.3 Output Signals of a Mechatronic System

### Digital-to-Analog Converters

The output command from the microcontroller is a binary value in bit, byte (8 bits), or word (16 bits) form. This digital signal is converted to analog using a digital-to-analog converter, or DAC. Let us examine converting an 8-bit value into a voltage level between 0 and 12 V. The most significant bit in the binary value to be converted (decimal 128) creates an analog value equal to half of the maximum output, or 6 V. The next digit produces an additional one fourth, or 3 V, the next an additional one eighth, and so forth. The sum of all these weighted output values represents the appropriate analog voltage. As was mentioned in a previous section, the maximum voltage value in the range is not obtainable, as the largest value generated is 255/256 of 12 V, or 11.953125 V. The smoothness of the signal representation depends on the number of bits accepted by the DAC and the range of the output required. Figure 3.7 demonstrates a simplified step function using a one-byte binary input and 12-V analog output.

### Actuator Output

Like sensors, actuators were first introduced in a previous section and will be described in detail in a later chapter of this handbook. The three common actuators that this section will review are switches, solenoids, and motors. Switches are simple state devices that control some activity, like turning on and off the furnace in a house. Types of switches include relays and solid-state devices. Solid-state devices include diodes, thyristors, bipolar transistors, field-effect transistors (FETs), and metal-oxide field-effect transistors (MOSFETs). A switch can also be used with a sensor, thus turning on or off the entire sensor, or a particular feature of a sensor.

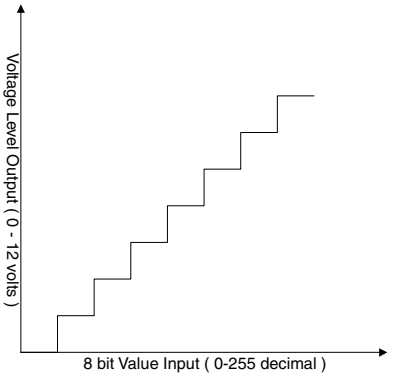



**FIGURE 3.7** DAC stepped output.

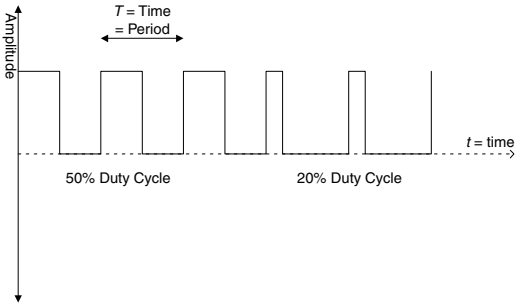



**FIGURE 3.8** Pulse width modulation.

Solenoids are devices containing a movable iron core that is activated by a current flow. The movement of this core can then control some form of hydraulic or pneumatic flow. Applications are many, including braking systems and industrial production of fluids. More information on solenoid actuators can be found in a later chapter. Motors are the last type of actuator that will be summarized here. There are three main types: direct current (DC), alternating current (AC), and stepper motors. DC motors may be controlled by a fixed DC voltage or by pulse width modulation (PWM). In a PWM signal, such as shown in Figure 3.8, a voltage is alternately turned on and off while changing (modulating) the width of the on-time signal, or duty cycle. AC motors are generally cheaper than DC motors, but require variable frequency drive to control the rotational speed. Stepper motors move by rotating a certain number of degrees in response to an input pulse.

## 3.4 Signal Conditioning

Signal conditioning is the modification of a signal to make it more useful to a system. Two important types of signal conditioning are, of course, the conversion between analog and digital, as described in the previous two sections. Other types of signal conditioning are briefly covered below.

### Sampling Rate

The rate at which data samples are taken obviously affects the speed at which the mechatronic system can detect a change in situation. There are several things to consider, however. For example, the response of a sensor may be limited in time or range. There is also the time required to convert the signal into a form usable by the microprocessor, the A to D conversion time. A third is the frequency of the signal being sampled. For voice digitalization, there is a very well-known sampling rate of 8000 samples per second.

This is a result of the Nyquist theorem, which states that the sampling rate, to be accurate, must be at least twice the maximum frequency being measured. The 8000 samples per second rate thus works well for converting human voice over an analog telephone system where the highest frequency is approximately 3400 Hz. Lastly, the clock speed of the microprocessor must also be considered. If the ADC and DAC are on the same board as the microprocessor, they will often share a common clock. The microprocessor clock, however, may be too fast for the ADC and DAC. In this case, a prescaler is used to divide the clock frequency to a level usable by the ADC and DAC.

## Filtering

Filtering is the attenuation (lessening) of certain frequencies from a signal. This process can remove noise from a signal and condition the line for better data transmission. Filters can be divided into analog and digital types, the analog filters being further divided into passive and active types. Analog passive filters use resistors, capacitors, and inductors. Analog active filters typically use operational amplifiers with resistors and capacitors. Digital filters may be implemented with software and/or hardware. The software component gives digital filters the feature of being easier to change.

Filters may also be differentiated by the type of frequencies they affect.

1. Low-pass filters allow lower set of frequencies to pass through, while high frequencies are attenuated. A simplistic example of this is shown in Figure 3.9.
2. High-pass filters, the opposite of low-pass, filter a lower frequency band while allowing higher frequencies to pass.
3. Band-pass filters allow a particular range of frequencies to pass; all others are attenuated.
4. Band-stop filters stop a particular range of frequencies while all others are allowed to pass.

There are many types and applications of filters. For example, William Ribbens in his book *Understanding Automotive Electronics* (Newnes, 1998) described a software low-pass filter (sometimes also called a lag filter) that averages the last 60 fuel tank level samples taken at 1 s intervals. The filtered data are then displayed on the vehicle instrument cluster. This type of filtering reduces large and quick fluctuations in the fuel gauge due to sloshing in the tank, and thus displays a more accurate value.

## Data Acquisition Boards

There is a special type of board that plugs into a slot in a desktop personal computer that can be used for many of the tasks above. It is called a data acquisition board, or DAQ board. This type of board can generate analog input and multiplex multiple input signals onto a single bus for transmission to the PC.

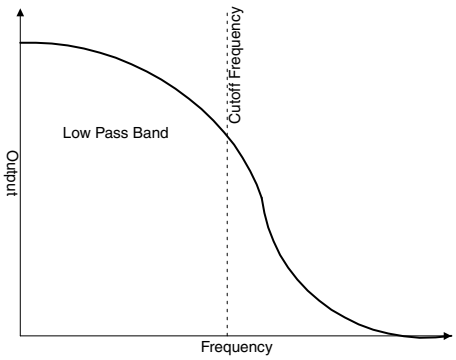



**FIGURE 3.9** Low-pass filter.

It can also come with signal conditioning hardware/software and an ADC. Some units have direct memory access (DMA), where the device writes the data directly into computer memory without using the microprocessor. While desktop PCs are not usually considered as part of a mechatronic system, the DAQ board can be very useful for instrumentation.

## 3.5 Microprocessor Control

### PID Control

A closed loop control system is one that determines a difference in the desired and actual condition (the error) and creates a correction control command to remove this error. PID control demonstrates three ways of looking at this error and correcting it. The first way is the P of PID, the proportional term. This term represents the control action made by the microcontroller in proportion to the error. In other words, the bigger the error, the bigger the correction. The I in PID is for the integral of the error over time. The integral term produces a correction that considers the time the error has been present. Stated in other words, the longer the error continues, the bigger the correction. Lastly, the D in PID stands for derivative. In the derivative term, the corrective action is related to the derivative or change of the error with respect to time. Stated in other words, the faster the error is changing, the bigger the correction. Control systems can use P, PI, PD, or PID in creating corrective actions. The problem generally is "tuning" the system by selecting the proper values in the terms.

### Programmable Logic Controllers

Any discussion of control systems and microprocessor control should start with the first type of "mechatronic" control, the programmable logic controller or PLC. A PLC is a simpler, more rugged microcontroller designed for environments like a factory floor. Input is usually from switches such as push buttons controlled by machine operators or position sensors. Timers can also be programmed in the PLC to run a particular process for a set amount of time. Outputs include lamps, solenoid valves, and motors, with the input–output interfacing done within the controller. A simple programming language used with a PLC is called ladder logic or ladder programming. Ladder logic is a graphical language showing logic as a combination of series (and's) and parallel (or's) blocks. Additional information can be found in Chapter 18 and in the book *Programmable Logic Controllers* by W. Bolton (Newnes 1996).

### Microprocessors

A full explanation of a microprocessor is found in section 5.8. For this discussion of microprocessors and control, we need only know a few of the component parts of computer architecture. RAM, or random access memory, is the set of memory locations the computer uses for fast temporary storage. The radio station presets selected by the driver (or passenger) in the car radio are stored in RAM. A small electrical current maintains these stored frequencies, so disconnection of the radio from the battery will result in their loss. ROM, or read only memory, is the static memory that contains the program to run the microcontroller. Thus the radio's embedded program will not be lost when the battery is disconnected. There are several types of ROM, including erasable programmable ROM (EPROM), electrically erasable programmable ROM (EEPROM), and flash memory (a newer type of EEPROM). These types will be explained later in this handbook. There are also special memory areas in a microprocessor called registers. Registers are very fast memory locations that temporarily store the address of the program instruction being executed, intermediate values needed to complete a calculation, data needed for comparison, and data that need to be input or output. Addresses and data are moved from one point to another in RAM, ROM, and registers using a bus, a set of lines transmitting data multiple bits simultaneously.

## 3.6 Microprocessor Numerical Control

### Fixed-Point Mathematics

The microprocessors in an embedded controller are generally quite small in comparison to a personal computer or computer workstation. Adding processing power in the form of a floating-point processor and additional RAM or ROM is not always an option. This means that sometimes the complex mathematical functions needed in a control system are not available. However, sometimes the values being sensed and computed, though real numbers, are of a reasonable range. Because of this situation there exists a special type of arithmetic whereby microcontrollers use integers in place of floating-point numbers to compute non-whole number (pseudo real) values.

There are several forms of fixed-point mathematics currently in use. The simplest form is based upon powers of 2, just like normal integers in binary. However, a virtual binary point is inserted into the integer to allow an approximation of real values to be stored as integers. A standard 8-bit unsigned integer is shown below along with its equivalent decimal value.

$$0001\ \ 0100 = (1 * 2^4) + (1 * 2^2) = (1 * 16) + (1 * 4) = 20$$

Suppose a virtual binary point is inserted between the two nibbles in the byte. There are now four bits left of the binary point with the standard positive powers of 2, and 4 bits right of the binary point with negative powers of 2. The same number now represents a real number in decimal.

$$0001\ \ 0100 = (1 * 2^0) + (1 * 2^{-2}) = (1 * 1) + (1 * 0.25) = 1.25$$

Obviously this method has shortcomings. The resolution of any fixed point number is limited to the power of 2 attached to the least significant bit on the right of the number, in this case $2^{-4}$ or 1/16 or 0.0625. Rounding is sometimes necessary. There is also a tradeoff in complexity, as the position of this virtual binary point must constantly be maintained when performing calculations. The savings in memory usage and processing time, however, often overcome these tradeoffs; so fixed-point mathematics can be very useful.

### Calibrations

The area of calibrating a system can sometimes take on an importance not foreseen when designing a mechatronic system. The use of calibrations, numerical and logical values kept in EEPROM or ROM, allow flexibility in system tuning and implementation. For example, if different microprocessor crystal speeds may be used in a mechatronic system, but real-time values are needed, a stored calibration constant of clock cycles per microsecond will allow this calculation to be affected. Thus, calibrations are often used as a gain, the value multiplied by some input in order to produce a scaled output.

Also, as mentioned above, calibrations are often used in the testing of a mechatronic system in order to change the "feel" of the product. A transmission control unit can use a set of calibrations on engine RPM, engine load, and vehicle speed to determine when to shift gears. This is often done with hysteresis, as the shift points moving from second gear to third gear as from third gear to second gear may differ.

## 3.7 Microprocessor Input–Output Control

### Polling and Interrupts

There are two basic methods for the microprocessor to control input and output. These are polling and interrupts. Polling is just that, the microprocessor periodically checking various peripheral devices to determine if input or output is waiting. If a peripheral device has some input or output that should be processed, a flag will be set. The problem is that a lot of processing time is wasted checking for inputs when they are not changing.

Servicing an interrupt is an alternative method to control inputs and outputs. In this method, a register in the microprocessor must have set an interrupt enable (IE) bit for a particular peripheral device. When an interrupt is initiated by the peripheral, a flag is set for the microprocessor. The interrupt request (IRQ) line will go active, and the microprocessor will service the interrupt. Servicing an interrupt means that the normal processing of the microprocessor is halted (i.e., interrupted) while the input/output is completed. In order to resume normal processing, the microprocessor needs to store the contents of its registers before the interrupt is serviced. This process includes saving all active register contents to a stack, a part of RAM designated for this purpose, in a process known as a push. After a push, the microprocessor can then load the address of the Interrupt Service Routine and complete the input/output. When that portion of code is complete, the contents of the stack are reloaded to the registers in an operation known as a Pop (or Pull) and normal processing resumes.

## Input and Output Transmission

Once the input or output is ready for transmission, there are several modes that can be used. First, data can be moved in either parallel or serial mode. Parallel mode means that multiple bits (e.g., 16 bits) move in parallel down a multiple pathway or bus from source to destination. Serial mode means that the bits move one at a time, in a series, down a single pathway. Parallel mode traffic is faster in that multiple bits are moving together, but the number of pathways is a limiting factor. For this reason parallel mode is usually used for components located close to one another while serial transmission is used if any distance is involved.

Serial data transmission can also be differentiated by being asynchronous or synchronous. Asynchronous data transmission uses separate clocks between the sender and receiver of data. Since these clocks are not synchronized, additional bits called start and stop bits are required to designate the boundaries of the bytes being sent. Synchronous data transmission uses a common or synchronized timing source. Start and stop bits are thus not needed, and overall throughput is increased.

A third way of differentiating data transmission is by direction. A simplex line is a one direction only pathway. Data from a sensor to the microcontroller may use simplex mode. Half-duplex mode allows two-way traffic, but only one direction at a time. This requires a form of flow control to avoid data transmission errors. Full-duplex mode allows two-way simultaneous transmission of data.

The agreement between sending and receiving units regarding the parameters of data transmission (including transmission speed) is known as handshaking.

## HC12 Microcontroller Input–Output Subsystems

There are four input–output subsystems on the Motorola HC12 microcontroller that can be used to exemplify the data transmission section above.

The serial communications interface (SCI) is an asynchronous serial device available on the HC12. It can be either polled or interrupt driven and is intended for communication between remote devices. Related to SCI is the serial peripheral interface (SPI). SPI is a synchronous serial interface. It is intended for communication between units that support SPI like a network of multiple microcontrollers. Because of the synchronization of timing that is required, SPI uses a system of master/slave relationships between microcontrollers.

The pulse width modulation (PWM) subsystem is often used for motor and solenoid control. Using registers that are mapped to both the PWM unit and the microprocessor, a PWM output can be commanded by setting values for the period and duty cycle in the proper registers. This will result in a particular on-time and off-time voltage command.

Last, the serial in-circuit debugger (SDI) allows the microcontroller to connect to a PC for checking and modifying embedded software.

## Microcontroller Network Systems

There is one last topic that should be mentioned in this section on inputs and outputs. Mechatronic systems often work with other systems in a network. Data and commands are thus transmitted from

one system to another. While there are many different protocols, both open and proprietary, that could be mentioned about this networking, two will serve our purposes. The first is the manufacturing automation protocol (MAP) that was developed by General Motors Corporation. This system is based on the ISO Open Systems Interconnection (OSI) model and is especially designed for computer integrated manufacturing (CIM) and multiple PLCs. The second is the controller area network (CAN). This standard for serial communications was developed by Robert Bosch GmbH for use among embedded systems in a car.

## 3.8 Software Control

### Systems Engineering

Systems engineering is the systems approach to the design and development of products and systems. As shown in Figure 3.10, a drawing that shows the relationships of the major engineering competencies with mechatronics, the systems engineering competency encompasses the mechanical, electrical, and software competencies. There are several important tasks for the systems engineers to perform, starting with requirements gathering and continuing through final product and system verification and validation. After requirements gathering and analysis, the systems engineers should partition requirements functionality between mechanical, electrical, and software components, in consultation with the three competencies involved. This is part of the implementation of concurrent engineering. As also shown by the figure, software is an equal partner in the development of a mechatronic system. It is not an add-on to the system and it is not free, the two opinions that were sometimes held in the past by engineering management. While the phrase "Hardware adds cost, software adds value" is not entirely true either, sometimes software engineers felt that their competency was not given equal weight with the traditional engineering disciplines. And one last comment—many mechatronic systems are safety related, such as an air bag system in a car. It is as important for the software to be as fault tolerant as the hardware.

### Software Engineering

Software engineering is concerned with both the final mechatronic "product" and the mechatronic development process. Two basic approaches are used with process, with many variations upon these approaches. One is called the "waterfall" method, where the process moves (falls) from one phase to another (e.g., analysis to design) with checkpoints along the way. The other method, the "spiral" approach, is often used when the requirements are not as well fixed. In this method there is prototyping, where the

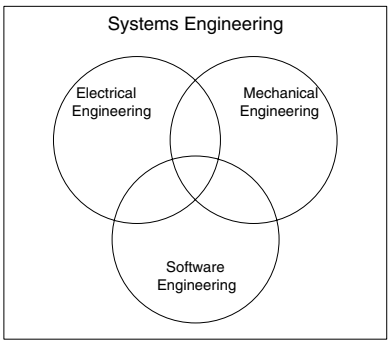



**FIGURE 3.10** Mechatronics engineering disciplines.

customers and/or systems engineers refine requirements as more information about the system becomes known. In either approach, once the requirements for the software portion of the mechatronic system are documented, the software engineers should further partition functionality as part of software design. Metrics as to development time, development cost, memory usage, and throughput should also be projected and recorded. Here is where the Software Engineering Institute's Capability Maturity Model (SEI CMM) levels can be used for guidance. It is a truism that software is almost never developed as easily as estimated, and that a system can remain at the "90% complete" level for most of the development life cycle. The first solution attempted to solve this problem is often assigning more software engineers onto the project. This does not always work, however, because of the learning curve of the new people, as stated by Frederick Brooks in his important book *The Mythical Man Month* (Addison-Wesley 1995).

## Software Design

Perhaps the most important part of the software design for a mechatronic system can be seen from the hierarchy in Figure 3.11. Ranging from requirements at the top to hardware at the bottom, this layering serves several purposes. The most important is that it separates mechatronic functionality from implementation. Quite simply, an upper layer should not be concerned with how a lower layer is actually performing a task. Each layer instead is directed by the layer above and receives a service or status from a layer below it. To cross more than one layer boundary is bad technique and can cause problems later in the process. Remember that this process abstraction is quite useful, for a mechatronic system has mechanical, electrical, and software parts all in concurrent development. A change in a sensor or actuator interface should only require a change at the layer immediately above, the driver layer. There is one last reason for using a hierarchical model such as this. In the current business climate, it is unlikely that the people working at the various layers will be collocated. Instead, it is not uncommon for development to be taking place in multiple locations in multiple countries. Without a crisp division of these layers, chaos can result.

For more information on these and many other topics in software engineering such as coupling, cohesion, and software reuse, please refer to Chapter 21 of this handbook, Roger Pressman's book *Software*

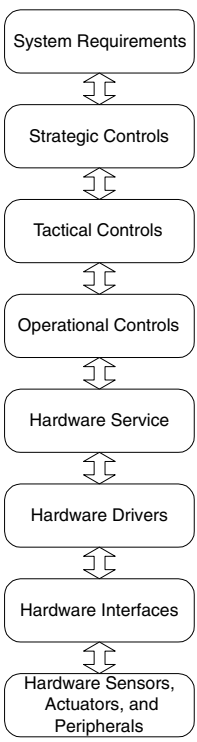



**FIGURE 3.11** Mechatronic software layering.

*Engineering: A Practitioner's Approach 5th Edition* (McGraw Hill 2000), and Steve McConnell's book *Code Complete* (Microsoft Press 1993).

## 3.9 Testing and Instrumentation

### Verification and Validation

Verification and validation are related tasks that should be completed throughout the life cycle of the mechatronic product or system. Boehm in his book *Software Engineering Economics* (Prentice-Hall 1988) describes verification as "building the product right" while validation is "building the right product." In other words, verification is the testing of the software and product to make sure that it is built to the design. Validation, on the other hand, is to make sure the software or product is built to the requirements from the customer. As mentioned, verification and validation are life cycle tasks, not tasks completed just before the system is set for production. One of the simplest and most useful techniques is to hold hardware and software validation and verification reviews. Validation design reviews of hardware and software should include the systems engineers who have the best understanding of the customer requirements. Verification hardware design and software code reviews, or peer reviews, are an excellent means of finding errors upstream in the development process. Managers may have to decide whether to allocate resources upstream, when the errors are easier to fix, or downstream, when the ramifications can be much more drastic. Consider the difference between a code review finding a problem in code, and having the author change it and recompile, versus finding a problem after the product has been sold and in the field, where an expensive product recall may be required.

### Debuggers

Edsgar Dijkstra, a pioneer in the development of programming as a discipline, discouraged the terms "bug" and "debug," and considered such terms harmful to the status of software engineering. They are, however, used commonly in the field. A debugger is a software program that allows a view of what is happening with the program code and data while the program is executing. Generally it runs on a PC that is connected to a special type of development microcontroller called an emulator. While debuggers can be quite useful in finding and correcting errors in code, they are not real-time, and so can actually create computer operating properly (COP) errors. However, if background debug mode (BDM) is available on the microprocessor, the debugger can be used to step through the algorithm of the program, making sure that the code is operating as expected. Intermediate and final variable values, especially those related to some analog input or output value, can be checked. Most debuggers allow multiple open windows, the setting of program execution break points in the code, and sometimes even the reflashing of the program into the microcontroller emulator. An example is the Noral debugger available for the Motorola HC12.

The software in the microcontroller can also check itself and its hardware. By programming in a checksum, or total, of designated portions of ROM and/or EEPROM, the software can check to make sure that program and data are correct. By alternately writing and reading 0x55 and 0xAA to RAM (the "checkerboard test"), the program can verify that RAM and the bus are operating properly. These startup tasks should be done with every product operation cycle.

### Logic Analyzer

A logic analyzer is a device for nonintrusive monitoring and testing of the microcontroller. It is usually connected to both the microcontroller and a simulator. While the microcontroller is running its program and processing data, the simulator is simulating inputs and displaying outputs of the system. A "trigger word" can be entered into the logic analyzer. This is a bit pattern that will be on one of the buses monitored by the logic analyzer. With this trigger, the bus traffic around that point of interest can be captured and stored in the memory of the analyzer. An inverse assembler in the analyzer allows the machine code on the

bus to be seen and analyzed in the form of the assembly level commands of the program. The analyzer can also capture the analog outputs of the microcontroller. This could be used to verify that the correct PWM duty cycle is being commanded. The simulator can introduce shorts or opens into the system, then the analyzer is used to see if the software correctly responds to the faults. The logic analyzer can also monitor the master loop of the system, making sure that the system completes all of its tasks within a designated time, e.g., 15 ms. An example of a logic analyzer is the Hewlett Packard HP54620.

## 3.10 Summary

This chapter introduced a number of topics regarding a mechatronic system. These topics included not just mechatronic input, output, and processing, but also design, development, and testing. Future chapters will cover all of this material in much greater detail.

# 4

# Microprocessor-Based Controllers and Microelectronics

Ondrej Novak
*Technical University Liberec*

Ivan Dolezal
*Technical University Liberec*



## 4.1 Introduction to Microelectronics

The field of microelectronics has changed dramatically during the last two decades and digital technology has governed most of the application fields in electronics. The design of digital systems is supported by thousands of different integrated circuits supplied by many manufacturers across the world. This makes both the design and the production of electronic products much easier and cost effective. The permanent growth of integrated circuit speed, scale of integration, and reduction of costs have resulted in digital circuits being used instead of classical analog solutions of controllers, filters, and (de)modulators.

The growth in computational power can be demonstrated with the following example. One single-chip microcontroller has the computational power equal to that of one 1992 vintage computer notebook. This single-chip microcontroller has the computational power equal to four 1981 vintage IBM personal computers, or to two 1972 vintage IBM 370 mainframe computers.

Digital integrated circuits are designed to be universal and are produced in large numbers. Modern integrated circuits have many upgraded features from earlier designs, which allow for "user-friendlier" access and control. As the parameters of Integrated circuits (ICs) influence not only the individually designed IC, but all the circuits that must cooperate with it, a roadmap of the future development of IC technology is updated every year. From this roadmap we can estimate future parameters of the ICs, and adapt our designs to future demands. The relative growth of the number of integrated transistors on a chip is relatively stable. In the case of memory elements, it is equal to approximately 1.5 times the current amount. In the case of other digital ICs, it is equal to approximately 1.35 times the current amount.

In digital electronics, we use quantities called logical values instead of the analog quantities of voltage and current. Logical variables usually correspond to the voltage of the signal, but they have only two values: log.1 and log.0. If a digital circuit processes a logical variable, a correct value is recognized because between the logical value voltages there is a gap (see Figure 4.1). We can arbitrarily improve the resolution of signals by simply using more bits.

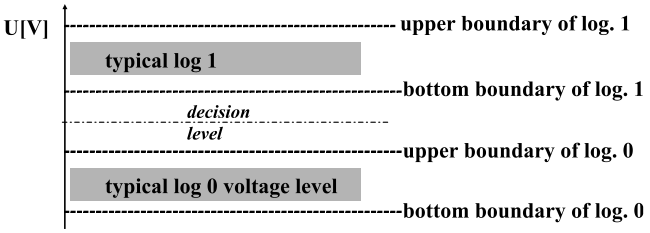



**FIGURE 4.1** Voltage levels and logical values correspondence.

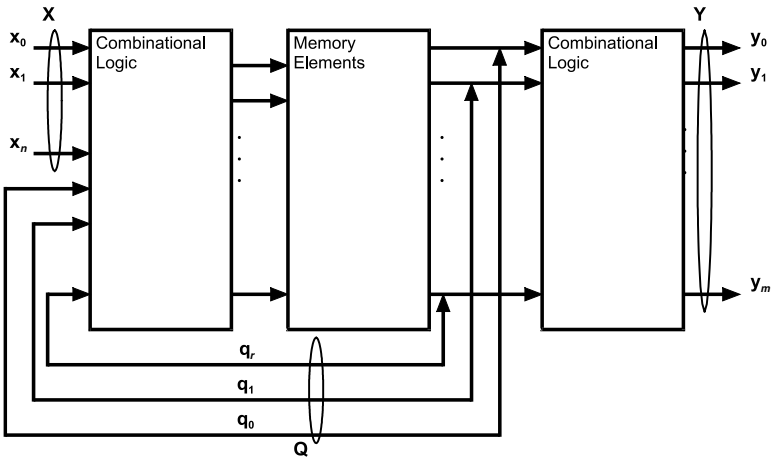



**FIGURE 4.2** A finite state automaton: *X*—input binary vector, *Y*—output binary vector, *Q*—internal state vector.

## 4.2 Digital Logic

Digital circuits are composed of logic gates, such as elementary electronic circuits operating in only two states. These gates operate in such a way that the resulting logical value corresponds to the resulting value of the Boolean algebra statements. This means that with the help of gates we can realize every logical and arithmetical operation. These operations are performed in combinational circuits for which the resulting value is dependent only on the actual state of the inputs variables. Of course, logic gates are not enough for automata construction. For creating an automaton, we also need some memory elements in which we capture the responses of the arithmetical and logical blocks.

A typical scheme of a digital finite state automaton is given in Figure 4.2. The automata can be constructed from standard ICs containing logic gates, more complex combinational logic blocks and registers, counters, memories, and other standard sequential ICs assembled on a printed circuit board. Another possibility is to use application specific integrated circuits (ASIC), either programmable or full custom, for a more advanced design. This approach is suitable for designs where fast hardware solutions are preferred. Another possibility is to use microcontrollers that are designed to serve as universal automata, which function can be specified by memory programming.

## 4.3 Overview of Control Computers

Huge, complex, and power-consuming single-room mainframe computers and, later, single-case minicomputers were primarily used for scientific and technical computing (e.g., in FORTRAN, ALGOL) and for database applications (e.g., in COBOL). The invention in 1971 of a universal central processing unit (CPU) in a single chip microprocessor caused a revolution in the computer technology. Beginning in 1981, multi-boxes (desktop or tower case, monitor, keyboard, mouse) or single-box (notebook)


**FIGURE 4.3** Example of a small mechatronic system: The ALAMBETA device for measurement of thermal properties of fabrics and plastic foils (manufactured by SENSORA, Czech Republic). It employs a unique measuring method using extra thin heat flow sensors, sample thickness measurement incorporated into a head drive, microprocessor control, and connection with a PC.

microcomputers became a daily-used personal tool for word processing, spreadsheet calculation, game playing, drawing, multimedia processing, and presentations. When connected in a local area network (LAN) or over the Internet, these "personal computers (PCs)" are able to exchange data and to browse the World Wide Web (WWW).

Besides these "visible" computers, many embedded microcomputers are hidden in products such as machines, vehicles, measuring instruments, telecommunication devices, home appliances, consumer electronic products (cameras, hi-fi systems, televisions, video recorders, mobile phones, music instruments, toys, air-conditioning). They are connected with sensors, user interfaces (buttons and displays), and actuators. Programmability of such controllers brings flexibility to the devices (function program choice), some kind of intelligence (fuzzy logic), and user-friendly action. It ensures higher reliability and easier maintenance, repairs, (auto)calibration, (auto)diagnostics, and introduces the possibility of their interconnection—mutual communication or hierarchical control in a whole plant or in a smart house. A photograph of an electrically operated instrument is given in Figure 4.3.

Embedded microcomputers are based on the Harvard architecture where code and data memories are split. Firmware (program code) is cross-compiled on a development system and then resides in a nonvolatile memory. In this way, a single main program can run immediately after a supply is switched on. Relatively expensive and shock sensitive mechanical memory devices (hard disks) and vacuum tube monitors have been replaced with memory cards or solid state disks (if an archive memory is essential) and LED segment displays or LCDs. A PC-like keyboard can be replaced by a device/function specifically labeled key set and/or common keys (arrows, Enter, Escape) completed with numeric keys, if necessary. Such key sets, auxiliary switches, large buttons, the main switch, and display can be located in water and dust resistant operator panels.

Progress in circuit integration caused fast development of microcontrollers in the last two decades. Code memory, data memory, clock generator, and a diverse set of peripheral circuits are integrated with the CPU (Figure 4.4) to insert such complete single-chip microcomputers into an application specific PCB.

Digital signal processors (DSPs) are specialized embedded microprocessors with some on-chip peripherals but with external ADC/DAC, which represent the most important input/output channel. DSPs have a parallel computing architecture and a fixed point or floating point instruction set optimized for typical signal processing operations such as discrete transformations, filtering, convolution, and coding. We can find DSPs in applications like sound processing/generation, sensor (e.g., vibration) signal analysis,

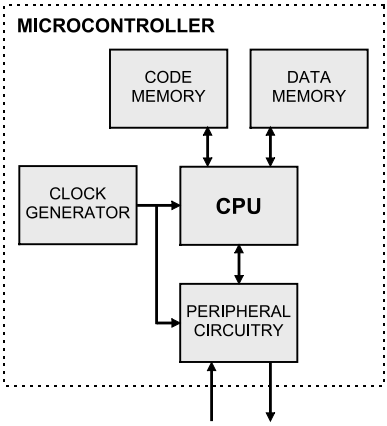



**FIGURE 4.4** Block diagram of a microcontroller.

telecommunications (e.g., bandpass filter and digital modulation/demodulation in mobile phones, communication transceivers, modems), and vector control of AC motors.

Mass production (i.e., low cost), wide-spread knowledge of operation, comprehensive access to software development and debugging tools, and millions of ready-to-use code lines make PCs useful for computing-intensive measurement and control applications, although their architecture and operating systems are not well suited for this purpose.

As a result of computer expansion, there exists a broad spectrum of computing/processing means from powerful workstations, top-end PCs and VXI systems (64/32 bits, over 1000 MFLOPS/MIPS, 1000 MB of memory, input power over 100 W, cost about $10,000), downwards to PC-based computer cards/modules (32 bits, 100–300 MFLOPS/MIPS, 10–100 MB, cost less than $1000). Microprocessor cards/modules (16/8 bits, 10–30 MIPS, 1 MB, cost about $100), complex microcontroller chips (16/8 bits, 10–30 MIPS, 10–100 KB, cost about $10), and simple 8-pin microcontrollers (8 bits, 1–5 MIPS, 1 KB, 10 mW, cost about $1) are also available for very little money.

## 4.4 Microprocessors and Microcontrollers

There is no strict border between microprocessors and microcontrollers because certain chips can access external code and/or data memory (microprocessor mode) and are equipped with particular peripheral components.

Some microcontrollers have an internal RC oscillator and do not need an external component. However, an external quartz or ceramic resonator or RC network is frequently connected to the built-in, active element of the clock generator. Clock frequency varies from 32 kHz (extra low power) up to 75 MHz. Another auxiliary circuit generates the reset signal for an appropriate period after a supply is turned on. Watchdog circuits generate chip reset when a periodic retriggering signal does not come in time due to a program problem. There are several modes of consumption reduction activated by program instructions.

Complexity and structure of the interrupt system (total number of sources and their priority level selection), settings of level/edge sensitivity of external sources and events in internal (i.e., peripheral) sources, and handling of simultaneous interrupt events appear as some of the most important criteria of microcontroller taxonomy.

Although 16- and 32-bit microcontrollers are engaged in special, demanding applications (servo-unit control), most applications employ 8-bit chips. Some microcontrollers can internally operate with a 16-bit or even 32-bit data only in fixed-point range—microcontrollers are not provided with floating point unit (FPU). New microcontroller families are built on RISC (Reduced Instruction Set) core executing due to pipelining one instruction per few clock cycles or even per each cycle.

One can find further differences in addressing modes, number of direct accessible registers, and type of code memory (ranging from 1 to 128 KB) that are important from the view of firmware development. Flash memory enables quick and even in-system programming (ISP) using 3–5 wires, whereas classical EPROM makes chips more expensive due to windowed ceramic packaging. Some microcontrollers have built-in boot and debug capability to load code from a PC into the flash memory using UART (Universal Asynchronous Receiver/Transmitter) and RS-232C serial line. OTP (One Time Programmable) EPROM or ROM appear effective for large production series. Data EEPROM (from 64 B to 4 KB) for calibration constants, parameter tables, status storage, and passwords that can be written by firmware stand beside the standard SRAM (from 32 B to 4 KB).

The range of peripheral components is very wide. Every chip has bidirectional I/O (input/output) pins associated in 8-bit ports, but they often have an alternate function. Certain chips can set an input decision level (TTL, MOS, or Schmitt trigger) and pull-up or pull-down current sources. Output drivers vary in open collector or tri-state circuitry and maximal currents.

At least one 8-bit timer/counter (usually provided with a prescaler) counts either external events (optional pulses from an incremental position sensor) or internal clocks, to measure time intervals, and periodically generates an interrupt or variable baud rate for serial communication. General purpose 16-bit counters and appropriate registers form either capture units to store the time of input transients or compare units that generate output transients as a stepper motor drive status or PWM (pulse width modulation) signal. A real-time counter (RTC) represents a special kind of counter that runs even in sleep mode. One or two asynchronous and optionally synchronous serial interfaces (UART/USART) communicate with a master computer while other serial interfaces like SPI, CAN, and $I^2C$ control other specific chips employed in the device or system.

Almost every microcontroller family has members that are provided with an A/D converter and a multiplexer of single-ended inputs. Input range is usually unipolar and equal to supply voltage or rarely to the on-chip voltage reference. The conversion time is given by the successive approximation principle of ADC, and the effective number of bits (ENOB) usually does not reach the nominal resolution 8, 10, or 12 bits.

There are other special interface circuits, such as field programmable gate array (FPGA), that can be configured as an arbitrary digital circuit.

Microcontroller firmware is usually programmed in an assembly language or in C language. Many software tools, including chip simulators, are available on websites of chip manufacturers or third-party companies free of charge. A professional integrated development environment and debugging hardware (in-circuit emulator) is more expensive (thousands of dollars). However, smart use of an inexpensive ROM simulator in a microprocessor system or a step-by-step development cycle using an ISP programmer of flash microcontroller can develop fairly complex applications.

## 4.5 Programmable Logic Controllers

A programmable logic controller (PLC) is a microprocessor-based control unit designed for an industrial installation (housing, terminals, ambient resistance, fault tolerance) in a power switchboard to control machinery or an industrial process. It consists of a CPU with memories and an I/O interface housed either in a compact box or in modules plugged in a frame and connected with proprietary buses. The compact box starts with about 16 I/O interfaces, while the module design can have thousands of I/O interfaces. Isolated inputs usually recognize industrial logic, 24 V DC or main AC voltage, while outputs are provided either with isolated solid state switches (24 V for solenoid valves and contactors) or with relays. Screw terminal boards represent connection facilities, which are preferred in PLCs to wire them to the controlled systems. I/O logical levels can be indicated with LEDs near to terminals.

Since PLCs are typically utilized to replace relays, they execute Boolean (bit, logical) operations and timer/counter functions (a finite state automaton). Analog I/O, integer or even floating point arithmetic, PWM outputs, and RTC are implemented in up-to-date PLCs. A PLC works by continually scanning a program, such as machine code, that is interpreted by an embedded microprocessor (CPU). The scan time is the time it takes to check the input status, to execute all branches (all individual rungs of a ladder

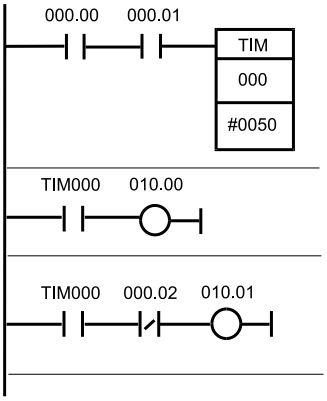



**FIGURE 4.5** Example of PLC ladder diagram: 000.xx/010.xx—address group of inputs/outputs, TIM000—timer delays 5 s. 000.00—normally open input contact, 000.02—normally closed input contact.

diagram) of the program using internal (state) bit variables if any, and to update the output status. The scan time is dependent on the complexity of the program (milliseconds or tens of msec). The next scan operation either follows the previous one immediately (free running) or starts periodically.

Programming languages for PLCs are described in IEC-1131-3 nomenclature:

LD—ladder diagram (see Figure 4.5)
IL—instruction list (an assembler)
SFC—sequential function chart (usually called by the proprietary name GRAFCET)
ST—structured text (similar to a high level language)
FBD—function block diagram

PLCs are programmed using cross-compiling and debugging tools running on a PC or with programming terminals (usually using IL), both connected with a serial link. Remote operator panels can serve as a human-to-machine interface. A new alternate concept (called SoftPLC) consists of PLC-like I/O modules controlled by an industrial PC, built in a touch screen operator panel.

## 4.6 Digital Communications

Intercommunication among mechatronics subsystems plays a key role in their engagement of applications, both of fixed and flexible configuration (a car, a hi-fi system, a fixed manufacturing line versus a flexible plant, a wireless pico-net of computer peripheral devices). It is clear that digital communication depends on the designers demands for the amount of transferred data, the distance between the systems, and the requirements on the degree of data reliability and security.

The signal is represented by alterations of amplitude, frequency, or phase. This is accomplished by changes in voltage/current in metallic wires or by electromagnetic waves, both in radiotransmission and infrared optical transmission (either "wireless" for short distances or optical fibers over fairly long distances). Data rate or bandwidth varies from 300 b/s (teleprinter), 3.4 kHz (phone), 144 kb/s (ISDN) to tens of Mb/s (ADSL) on a metallic wire (subscriber line), up to 100 Mb/s on a twisted pair (LAN), about 30–100 MHz on a microwave channel, 1 GHz on a coaxial cable (trunk cable network, cable TV), and up to tens of Gb/s on an optical cable (backbone network).

Data transmission employs complex methods of digital modulation, data compression, and data protection against loss due to noise interference, signal distortion, and dropouts. Multilayer standard protocols (ISO/OSI 7-layer reference model or Internet 4-layer group of protocols including well-known TCP/IP), "partly hardware, partly software realized," facilitate an understanding between communication systems. They not only establish connection on a utilizable speed, check data transfer, format and compress data, but can make communication transparent for an application. For example, no difference

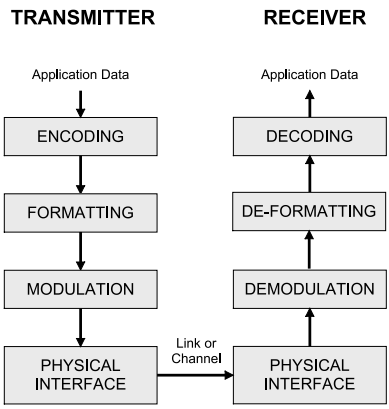



**FIGURE 4.6** Example of multilayer communication.

can be seen between local and remote data sources. An example of a multilayer communication concept is depicted in Figure 4.6.

Depending on the number of users, the communication is done either point-to-point (RS-232C from PC COM port to an instrument), point-to-multipoint (buses, networks), or even as a broadcasting (radio). Data are transferred using either switched connection (telephone network) or packet switching (computer networks, ATM). Bidirectional transmission can be full duplex (phone, RS-232C) or semi-duplex (most of digital networks). Concerning the link topology, a star connection or a tree connection employs a device ("master") mastering communication in the main node(s). A ring connection usually requires Token Passing method and a bus communication is controlled with various methods such as Master-Slave pooling, with or without Token Passing, or by using an indeterministic access (CSMA/CD in Ethernet).

An LPT PC port, SCSI for computer peripherals, and GPIB (IEEE-488) for instrumentation serve as examples of parallel (usually 8-bit) communication available for shorter distances (meters). RS-232C, RS-485, $I^2C$, SPI, USB, and Firewire (IEEE-1394) represent serial communication, some of which can bridge long distance (up to 1 km). Serial communication can be done either asynchronously using start and stop bits within transfer frame or synchronously using included synchronization bit patterns, if necessary. Both unipolar and bipolar voltage levels are used to drive either unbalanced lines (LPT, GPIB vs. RS-232C) or balanced twisted-pair lines (CAN vs. RS-422, RS-485).

# 5

# An Introduction to Micro- and Nanotechnology

Michael Goldfarb
*Vanderbilt University*

Alvin Strauss
*Vanderbilt University*

Eric J. Barth
*Vanderbilt University*



## 5.1 Introduction

Originally arising from the development of processes for fabricating microelectronics, micro-scale devices are typically classified according not only to their dimensional scale, but their composition and manufacture. Nanotechnology is generally considered as ranging from the smallest of these micro-scale devices down to the assembly of individual molecules to form molecular devices. These two distinct yet overlapping fields of microelectromechanical systems (MEMS) and nanosystems or nanotechnology share a common set of engineering design considerations unique from other more typical engineering systems. Two major factors distinguish the existence, effectiveness, and development of micro-scale and nano-scale transducers from those of conventional scale. The first is the physics of scaling and the second is the suitability of manufacturing techniques and processes. The former is governed by the laws of physics and is thus a fundamental factor, while the latter is related to the development of manufacturing technology, which is a significant, though not fundamental, factor. Due to the combination of these factors, effective micro-scale transducers can often not be constructed as geometrically scaled-down versions of conventional-scale transducers.

### The Physics of Scaling

The dominant forces that influence micro-scale devices are different from those that influence their conventional-scale counterparts. This is because the size of a physical system bears a significant influence on the physical phenomena that dictate the dynamic behavior of that system. For example, larger-scale systems are influenced by inertial effects to a much greater extent than smaller-scale systems, while smaller systems are influenced more by surface effects. As an example, consider small insects that can stand on the surface of still water, supported only by surface tension. The same surface tension is present when

humans come into contact with water, but on a human scale the associated forces are typically insignificant. The world in which humans live is governed by the same forces as the world in which these insects live, but the forces are present in very different proportions. This is due in general to the fact that inertial forces typically act in proportion to volume, and surface forces typically in proportion to surface area. Since volume varies with the third power of length and area with the second, geometrically similar but smaller objects have proportionally more area than larger objects.

Exact scaling relations for various types of forces can be obtained by incorporating dimensional analysis techniques [1–5]. Inertial forces, for example, can be dimensionally represented as $F_i = \rho L^3 \ddot{x}$, where $F_i$ is a generalized inertia force, $\rho$ is the density of an object, $L$ is a generalized length, and $x$ is a displacement. This relationship forms a single dimensionless group, given by

$$\prod = \frac{F_i}{\rho L^3 \ddot{x}}$$

Scaling with geometric and kinematic similarity can be expressed as

$$\frac{L_s}{L_o} = \frac{x_s}{x_o} = N, \qquad \frac{t_s}{t_o} = 1$$

where $L$ represents the length scale, $x$ the kinematic scale, $t$ the time scale, the subscript $o$ the original system, and the $s$ represents the scaled system. Since physical similarity requires that the dimensionless group ($\Pi$) remain invariant between scales, the force relationship is given by $F_s/F_o = N^4$, assuming that the intensive property (density) remains invariant (i.e., $\rho_s = \rho_o$). An inertial force thus scales as $N^4$, where $N$ is the geometric scaling factor. Alternately stated, for an inertial system that is geometrically smaller by a factor of $N$, the force required to produce an equivalent acceleration is smaller by a factor of $N^4$. A similar analysis shows that viscous forces, dimensionally represented by $F_v = \mu L \dot{x}$, scale as $N^2$, assuming the viscosity $\mu$ remains invariant, and elastic forces, dimensionally represented by $F_e = ELx$, scale as $N^2$, assuming the elastic modulus $E$ remains invariant. Thus, for a geometrically similar but smaller system, inertial forces will become considerably less significant with respect to viscous and elastic forces.

## General Mechanisms of Electromechanical Transduction

The fundamental mechanism for both sensing and actuation is energy transduction. The primary forms of physical electromechanical transduction can be grouped into two categories. The first is multicomponent transduction, which utilizes "action at a distance" behavior between multiple bodies, and the second is deformation-based or solid-state transduction, which utilizes mechanics-of-material phenomena such as crystalline phase changes or molecular dipole alignment. The former category includes electromagnetic transduction, which is typically based upon the Lorentz equation and Faraday's law, and electrostatic interaction, which is typically based upon Coulomb's law. The latter category includes piezoelectric effects, shape memory alloys, and magnetostrictive, electrostrictive, and photostrictive materials. Although materials exhibiting these properties are beginning to be seen in a limited number of research applications, the development of micro-scale systems is currently dominated by the exploitation of electrostatic and electromagnetic interactions. Due to their importance, electrostatic and electromagnetic transduction is treated separately in the sections that follow.

## Sensor and Actuator Transduction Characteristics

Characteristics of concern for both microactuator and microsensor technology are repeatability, the ability to fabricate at a small scale, immunity to extraneous influences, sufficient bandwidth, and if possible, linearity. Characteristics typically of concern specifically for microactuators are achievable force, displacement, power, bandwidth (or speed of response), and efficiency. Characteristics typically of concern specifically for microsensors are high resolution and the absence of drift and hysteresis.

## 5.2 Microactuators

### Electrostatic Actuation

The most widely utilized multicomponent microactuators are those based upon electrostatic transduction. These actuators can also be regarded as a variable capacitance type, since they operate in an analogous mode to variable reluctance type electromagnetic actuators (e.g., variable reluctance stepper motors). Electrostatic actuators have been developed in both linear and rotary forms. The two most common configurations of the linear type of electrostatic actuators are the normal-drive and tangential or comb-drive types, which are illustrated in Figures 5.1 and 5.2, respectively. Note that both actuators are suspended by flexures, and thus the output force is equal to the electrostatic actuation force minus the elastic force required to deflect the flexure suspension. The normal-drive type of electrostatic microactuator operates in a similar fashion to a condenser microphone. In this type of drive configuration, the actuation force is given by

$$F_x = \frac{\varepsilon A v^2}{2x^2}$$

where $A$ is the total area of the parallel plates, $\varepsilon$ is the permittivity of air, $v$ is the voltage across the plates, and $x$ is the plate separation. The actuation force of the comb-drive configuration is given by

$$F_x = \frac{\varepsilon w v^2}{2d}$$

where $w$ is the width of the plates, $\varepsilon$ is the permittivity of air, $v$ is the voltage across the plates, and $d$ is the plate separation. Dimensional examination of both relations indicates that force is independent of geometric and kinematic scaling, that is, for an electrostatic actuator that is geometrically and kinematically reduced by a factor of $N$, the force produced by that actuator will be the same. Since forces associated with most other physical phenomena are significantly reduced at small scales, micro-scale electrostatic forces become significant relative to other forces. Such an observation is clearly demonstrated by the fact that all intermolecular forces are electrostatic in origin, and thus the strength of all materials is a result of electrostatic forces [6].

The maximum achievable force of multicomponent electrostatic actuators is limited by the dielectric breakdown of air, which occurs in dry air at about $0.8 \times 10^6$ V/m. Fearing [7] estimates that the upper limit for force generation in electrostatic actuation is approximately 10 N/cm$^2$. Since electrostatic drives

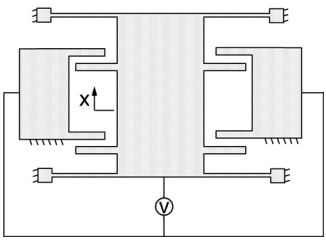



**FIGURE 5.1** Schematic of a normal-drive electrostatic actuator.

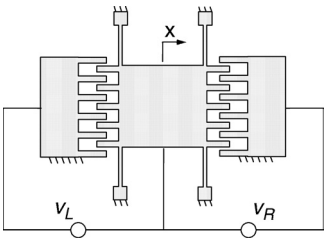



**FIGURE 5.2** Comb-drive electrostatic actuator. Energizing an electrode provides motion toward that electrode.

do not have any significant actuation dynamics, and since the inertia of the moving member is usually small, the actuator bandwidth is typically quite large, on the order of a kilohertz.

The maximum achievable stroke for normal configuration actuators is limited by the elastic region of the flexure suspension and additionally by the dependence of actuation force on plate separation, as given by the above stated equations. According to Fearing, a typical stroke for a surface micromachined normal configuration actuator is on the order of a couple of microns. The achievable displacement can be increased by forming a stack of normal-configuration electrostatic actuators in series, as proposed by Bobbio et al. [8,9].

The typical stroke of a surface micromachined comb actuator is on the order of a few microns, though sometimes less. The maximum achievable stroke in a comb drive is limited primarily by the mechanics of the flexure suspension. The suspension should be compliant along the direction of actuation to enable increased displacement, but must be stiff orthogonal to this direction to avoid parallel plate contact due to misalignment. These modes of behavior are unfortunately coupled, so that increased compliance along the direction of motion entails a corresponding increase in the orthogonal direction. The net effect is that increased displacement requires increased plate separation, which results in decreased overall force.

The most common configurations of rotary electrostatic actuators are the variable capacitance motor and the wobble or harmonic drive motor, which are illustrated in Figures 5.3 and 5.4, respectively. Both motors operate in a similar manner to the comb-drive linear actuator. The variable capacitance motor is characterized by high-speed low-torque operation. Useful levels of torque for most applications therefore require some form of significant micromechanical transmission, which do not presently exist. The rotor of the wobble motor operates by rolling along the stator, which provides an inherent harmonic-drive-type transmission and thus a significant transmission ratio (on the order of several hundred times). Note that the rotor must be well insulated to roll along the stator without electrical contact. The drawback to this approach is that the rotor motion is not concentric with respect to the stator, which makes the already difficult problem of coupling a load to a micro-shaft even more difficult.

Examples of normal type linear electrostatic actuators are those by Bobbio et al. [8,9] and Yamaguchi et al. [10]. Examples of comb-drive electrostatic actuators are those by Kim et al. [11] and Matsubara et al. [12], and a larger-scale variation by Niino et al. [13]. Examples of variable capacitance rotary electrostatic motors are those by Huang et al. [14], Mehragany et al. [15], and Trimmer and Gabriel [16].

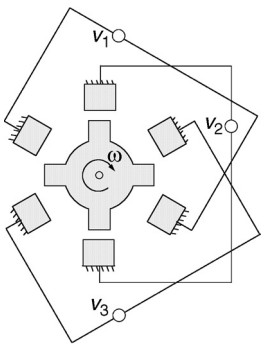



**FIGURE 5.3** Variable capacitance type electrostatic motor. Opposing pairs of electrodes are energized sequentially to rotate the rotor.

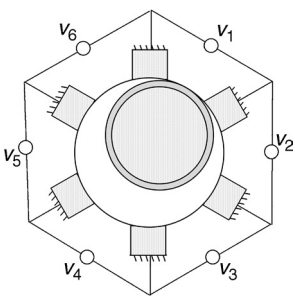



**FIGURE 5.4** Harmonic drive type electrostatic motor. Adjacent electrodes are energized sequentially to roll the (insulated) rotor around the stator.

Examples of harmonic-drive motors are those by Mehragany et al. [17,18], Price et al. [19], Trimmer and Jebens [20,21], and Furuhata et al. [22]. Electrostatic microactuators remain a subject of research interest and development, and as such are not yet available on the general commercial market.

## Electromagnetic Actuation

Electromagnetic actuation is not as omnipresent at the micro-scale as at the conventional-scale. This probably is due in part to early skepticism regarding the scaling of magnetic forces, and in part to the fabrication difficulty in replicating conventional-scale designs. Most electromagnetic transduction is based upon a current carrying conductor in a magnetic field, which is described by the Lorentz equation:

$$dF = Idl \times B$$

where $F$ is the force on the conductor, $I$ is the current in the conductor, $l$ is the length of the conductor, and $B$ is the magnetic flux density. In this relation, the magnetic flux density is an intensive variable and thus (for a given material) does not change with scale. Scaling of current, however, is not as simple. The resistance of wire is given by

$$R = \frac{\rho l}{A}$$

where $\rho$ is the resistivity of the wire (an intensive variable), $l$ is the length, and $A$ the cross-sectional area. If a wire is geometrically decreased in size by a factor of $N$, its resistance will increase by a factor of $N$. Since the power dissipated in the wire is $I^2R$, assuming the current remains constant implies that the power dissipated in the geometrically smaller wire will increase by a factor of $N$. Assuming the maximum power dissipation for a given wire is determined by the surface area of the wire, a wire that is smaller by a factor of $N$ will be able to dissipate a factor of $N^2$ less power. Constant current is therefore a poor assumption. A better assumption is that maximum current is limited by maximum power dissipation, which is assumed to depend upon surface area of the wire. Since a wire smaller by a factor of $N$ can dissipate a factor of $N^2$ less power, the current in the smaller conductor would have to be reduced by a factor of $N^{3/2}$. Incorporating this into the scaling of the Lorentz equation, an electromagnetic actuator that is geometrically smaller by a factor of $N$ would exert a force that is smaller by a factor of $N^{5/2}$. Trimmer and Jebens have conducted a similar analysis, and demonstrated that electromagnetic forces scale as $N^2$ when assuming constant temperature rise in the wire, $N^{5/2}$ when assuming constant heat (power) flow (as previously described), and $N^3$ when assuming constant current density [23,24]. In any of these cases, the scaling of electromagnetic forces is not nearly as favorable as the scaling of electrostatic forces. Despite this, electromagnetic actuation still offers utility in microactuation, and most likely scales more favorably than does inertial or gravitational forces.

Lorentz-type approaches to microactuation utilize surface micromachined micro-coils, such as the one illustrated in Figure 5.5. One configuration of this approach is represented by the actuator of Inoue et

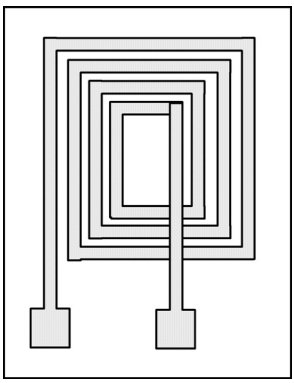


**FIGURE 5.5** Schematic of surface micromachined microcoil for electromagnetic actuation.

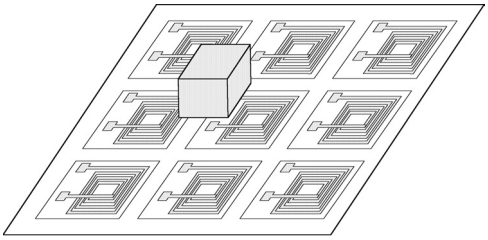


**FIGURE 5.6** Microcoil array for planar positioning of a permanent micromagnet, as described by Inoue et al. [25]. Each coil produces a field, which can either attract or repel the permanent magnet, as determined by the direction of current. The magnet does not levitate, but rather slides on the insulated surface.

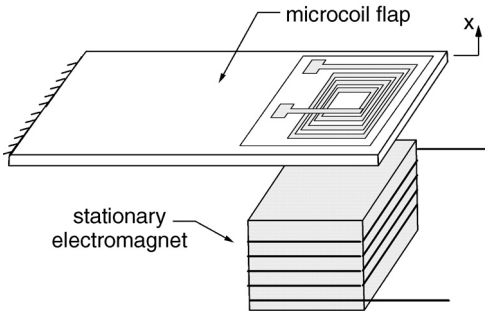



**FIGURE 5.7** Cantilevered microcoil flap as described by Liu et al. [26]. The interaction between the energized coil and the stationary electromagnet deflects the flap upward or downward, depending on the direction of current through the microcoil.

al. [25], which utilizes current control in an array of microcoils to position a permanent micro-magnet in a plane, as illustrated in Figure 5.6. Another Lorentz-type approach is illustrated by the actuator of Liu et al. [26], which utilizes current control of a cantilevered microcoil flap in a fixed external magnetic field to effect deflection of the flap, as shown in Figure 5.7. Liu reported deflections up to 500 $\mu$m and a bandwidth of approximately 1000 Hz [26]. Other examples of Lorentz-type nonrotary actuators are those by Shinozawa et al. [27], Wagner and Benecke [28], and Yanagisawa et al. [29]. A purely magnetic approach (i.e., not fundamentally electromagnetic) is the work of Judy et al. [30], which in essence manipulates a flexure-suspended permanent micromagnet by controlling an external magnetic field.

Ahn et al. [31] and Guckel et al. [32] have both demonstrated planar rotary variable-reluctance type electromagnetic micromotors. A variable reluctance approach is advantageous because the rotor does not require commutation and need not be magnetic. The motor of Ahn et al. incorporates a 12-pole stator and 10-pole rotor, while the motor of Guckel et al. utilizes a 6-pole stator and 4-pole rotor. Both incorporate rotors of approximately 500 $\mu$m diameter. Guckel reports (no load) rotor speeds above 30,000 rev/min, and Ahn estimates maximum stall torque at 1.2 $\mu$N m. As with electrostatic microactuators, microfabricated electromagnetic actuators likewise remain a subject of research interest and development and as such are not yet available on the general commercial market.

## 5.3 Microsensors

Since microsensors do not transmit power, the scaling of force is not typically significant. As with conventional-scale sensing, the qualities of interest are high resolution, absence of drift and hysteresis, achieving a sufficient bandwidth, and immunity to extraneous effects not being measured.

Microsensors are typically based on either measurement of mechanical strain, measurement of mechanical displacement, or on frequency measurement of a structural resonance. The former two types

are in essence analog measurements, while the latter is in essence a binary-type measurement, since the sensed quantity is typically the frequency of vibration. Since the resonant-type sensors measure frequency instead of amplitude, they are generally less susceptible to noise and thus typically provide a higher resolution measurement. According to Guckel et al., resonant sensors provide as much as one hundred times the resolution of analog sensors [33]. They are also, however, more complex and are typically more difficult to fabricate.

The primary form of strain-based measurement is piezoresistive, while the primary means of displacement measurement is capacitive. The resonant sensors require both a means of structural excitation as well as a means of resonant frequency detection. Many combinations of transduction are utilized for these purposes, including electrostatic excitation, capacitive detection, magnetic excitation and detection, thermal excitation, and optical detection.

## Strain

Many microsensors are based upon strain measurement. The primary means of measuring strain is via piezoresistive strain gages, which is an analog form of measurement. Piezoresistive strain gages, also known as semiconductor gages, change resistance in response to a mechanical strain. Note that piezoelectric materials can also be utilized to measure strain. Recall that mechanical strain will induce an electrical charge in a piezoelectric ceramic. The primary problem with using a piezoelectric material, however, is that since measurement circuitry has limited impedance, the charge generated from a mechanical strain will gradually leak through the measurement impedance. A piezoelectric material therefore cannot provide reliable steady-state signal measurement. In constrast, the change in resistance of a piezoresistive material is stable and easily measurable for steady-state signals. One problem with piezoresistive materials, however, is that they exhibit a strong strain-temperature dependence, and so must typically be thermally compensated.

An interesting variation on the silicon piezoresistor is the resonant strain gage proposed by Ikeda et al., which provides a frequency-based form of measurement that is less susceptible to noise [34]. The resonant strain gage is a beam that is suspended slightly above the strain member and attached to it at both ends. The strain gage beam is magnetically excited with pulses, and the frequency of vibration is detected by a magnetic detection circuit. As the beam is stretched by mechanical strain, the frequency of vibration increases. These sensors provide higher resolution than typical piezoresistors and have a lower temperature coefficient. The resonant sensors, however, require a complex three-dimensional fabrication technique, unlike the typical piezoresistors which require only planar techniques.

## Pressure

One of the most commercially successful microsensor technologies is the pressure sensor. Silicon micromachined pressure sensors are available that measure pressure ranges from around one to several thousand kPa, with resolutions as fine as one part in ten thousand. These sensors incorporate a silicon micromachined diaphragm that is subjected to fluid (i.e., liquid or gas) pressure, which causes dilation of the diaphragm. The simplest of these utilize piezoresistors mounted on the back of the diaphragm to measure deformation, which is a function of the pressure. Examples of these devices are those by Fujii et al. [35] and Mallon et al. [36]. A variation of this configuration is the device by Ikeda et al. Instead of a piezoresistor to measure strain, an electromagnetically driven and sensed resonant strain gage, as discussed in the previous section, is utilized [37]. Still another variation on the same theme is the capacitive measurement approach, which measures the capacitance between the diaphragm and an electrode that is rigidly mounted and parallel to the diaphragm. An example of this approach is by Nagata et al. [38]. A more complex approach to pressure measurement is that by Stemme and Stemme, which utilizes resonance of the diaphragm to detect pressure [39]. In this device, the diaphragm is capacitively excited and optically detected. The pressure imposes a mechanical load on the diaphragm, which increases the stiffness and, in turn, the resonant frequency.

## Acceleration

Another commercially successful microsensor is the silicon microfabricated accelerometer, which in various forms can measure acceleration ranges from well below one to around a thousand meters per square second (i.e., sub-*g* to several hundred *g*'s), with resolutions of one part in 10,000. These sensors incorporate a micromachined suspended proof mass that is subjected to an inertial force in response to an acceleration, which causes deflection of the supporting flexures. One means of measuring the deflection is by utilizing piezoresistive strain gages mounted on the flexures. The primary disadvantage to this approach is the temperature sensitivity of the piezoresistive gages. An alternative to measuring the deflection of the proof mass is via capacitive sensing. In these devices, the capacitance is measured between the proof mass and an electrode that is rigidly mounted and parallel. Examples of this approach are those by Boxenhorn and Greiff [40], Leuthold and Rudolf [41], and Seidel et al. [42]. Still another means of measuring the inertial force on the proof mass is by measuring the resonant frequency of the supporting flexures. The inertial force due to acceleration will load the flexure, which will alter its resonant frequency. The frequency of vibration is therefore a measure of the acceleration. These types of devices utilize some form of transduction to excite the structural resonance of the supporting flexures, and then utilize some other measurement technique to detect the frequency of vibration. Examples of this type of device are those by Chang et al. [43], which utilize electrostatic excitation and capacitive detection, and by Satchell and Greenwood [44], which utilize thermal excitation and piezoresistive detection. These types of accelerometers entail additional complexity, but typically offer improved measurement resolution. Still another variation of the micro-accelerometer is the force-balanced type. This type of device measures position of the proof mass (typically by capacitive means) and utilizes a feedback loop and electrostatic or electromagnetic actuation to maintain zero deflection of the mass. The acceleration is then a function of the actuation effort. These devices are characterized by a wide bandwidth and high sensitivity, but are typically more complex and more expensive than other types. Examples of force-balanced devices are those by Chau et al. [45], and Kuehnel and Sherman [46], both of which utilize capacitive sensing and electrostatic actuation.

## Force

Silicon microfabricated force sensors incorporate measurement approaches much like the microfabricated pressure sensors and accelerometers. Various forms of these force sensors can measure forces ranging on the order of millinewtons to newtons, with resolutions of one part in 10,000. Mechanical sensing typically utilizes a beam or a flexure support which is elastically deflected by an applied force, thereby transforming force measurement into measurement of strain or displacement, which can be accomplished by piezoresistive or capacitive means. An example of this type of device is that of Despont et al., which utilizes capacitive measurement [47]. Higher resolution devices are typically of the resonating beam type, in which the applied force loads a resonating beam in tension. Increasing the applied tensile load results in an increase in resonant frequency. An example of this type of device is that of Blom et al. [48].

## Angular Rate Sensing (Gyroscopes)

A conventional-scale gyroscope utilizes the spatial coupling of the angular momentum-based gyroscopic effect to measure angular rate. In these devices, a disk is spun at a constant high rate about its primary axis, so that when the disk is rotated about an axis not colinear with the primary (or spin) axis, a torque results in an orthogonal direction that is proportional to the angular velocity. These devices are typically mounted in gimbals with low-friction bearings, incorporate motors that maintain the spin velocity, and utilize strain gages to measure the gyroscopic torque (and thus angular velocity). Such a design would not be appropriate for a microsensor due to several factors, some of which include the diminishing effect of inertia (and thus momentum) at small scales, the lack of adequate bearings, the lack of appropriate micromotors, and the lack of an adequate three-dimensional microfabrication processes. Instead, microscale angular rate sensors are of the vibratory type, which incorporate Coriolis-type effects rather than

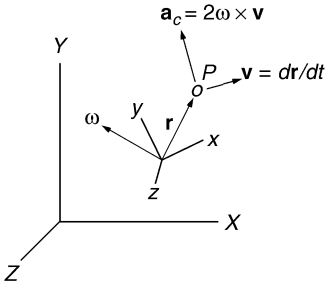



**FIGURE 5.8** Illustration of Coriolis acceleration, which results from translation within a reference frame that is rotating with respect to an inertial reference frame.

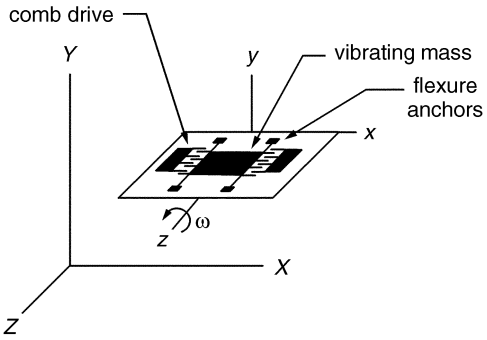



**FIGURE 5.9** Schematic of a vibratory gyroscope.

the angular momentum-based gyroscopic mechanics of conventional-scale devices. A Coriolis acceleration results from linear translation within a coordinate frame that is rotating with respect to an inertial reference frame. In particular, if the particle in Figure 5.8 is moving with a velocity $v$ within the frame $xyz$, and if the frame $xyz$ is rotating with an angular velocity of $\omega$ with respect to the inertial reference frame $XYZ$, then a Coriolis acceleration will result equal to $\mathbf{a}_c = 2\omega \times v$. If the object has a mass $m$, a Coriolis inertial force will result equal to $\mathbf{F}_c = -2m\omega \times v$ (minus sign because direction is opposite $\mathbf{a}_c$). A vibratory gyroscope utilizes this effect as illustrated in Figure 5.9. A flexure-suspended inertial mass is vibrated in the $x$-direction, typically with an electrostatic comb drive. An angular velocity about the $z$-axis will generate a Coriolis acceleration, and thus force, in the $y$-direction. If the "external" angular velocity is constant and the velocity in the $x$-direction is sinusoidal, then the resulting Coriolis force will be sinusiodal, and the suspended inertial mass will vibrate in the $y$-direction with an amplitude proportional to the angular velocity. The motion in the $y$-direction, which is typically measured capacitively, is thus a measure of the angular rate. Examples of these types of devices are those by Bernstein et al. [49] and Oh et al. [50]. Note that though vibration is an essential component of these devices, they are not technically resonant sensors, since they measure amplitude of vibration rather than frequency.

## 5.4 Nanomachines

Nanomachines are devices that range in size from the smallest of MEMS devices down to devices assembled from individual molecules [51]. This section briefly introduces energy sources, structural hierarchy, and the projected future of the assembly of nanomachines. Built from molecular components performing individual mechanical functions, the candidates for energy sources to actuate nanomachines are limited to those that act on a molecular scale. Regarding manufacture, the assembly of nanomachines is by nature a one-molecule-at-a-time operation. Although microscopy techniques are currently used for the assembly of nanostructures, self-assembly is seen as a viable means of mass production.

In a molecular device a discrete number of molecular components are combined into a supramolecular structure where each discrete molecular component performs a single function. The combined action of these individual molecules causes the device to operate and perform its various functions. Molecular devices require an energy source to operate. This energy must ultimately be used to activate the component molecules in the device, and so the energy must be chemical in nature. The chemical energy can be obtained by adding hydrogen ions, oxidants, etc., by inducing chemical reactions by the impingement of light, or by the actions of electrical current. The latter two means of energy activation, photochemical and electrochemical energy sources, are preferred since they not only provide energy for the operation of the device, but they can also be used to locate and control the device. Additionally, such energy transduction can be used to transmit data to report on the performance and status of the device. Another reason for the preference for photochemical- and electrochemical-based molecular devices is that, as these devices are required to operate in a cyclic manner, the chemical reactions that drive the system must be reversible. Since photochemical and electrochemical processes do not lead to the accumulation of products of reaction, they readily lend themselves to application in nanodevices.

Molecular devices have recently been designed that are capable of motion and control by photochemical methods. One device is a molecular plug and socket system, and another is a piston-cylinder system [51]. The construction of such supramolecular devices belongs to the realm of the chemist who is adept at manipulating molecules.

As one proceeds upwards in size to the next level of nanomachines, one arrives at devices assembled from (or with) single-walled carbon nanotubes (SWNTs) and/or multi-walled carbon nanotubes (MWNTs) that are a few nanometers in diameter. We will restrict our discussion to carbon nanotubes (CNTs) even though there is an expanding database on nanotubes made from other materials, especially bismuth. The strength and versatility of CNTs make them superior tools for the nanomachine design engineer. They have high electrical conductivity with current carrying capacity of a billion amperes per square centimeter. They are excellent field emitters at low operating voltages. Moreover, CNTs emit light coherently and this provides for an entire new area of holographic applications. The elastic modulus of CNTs is the highest of all materials known today [52]. These electrical properties and extremely high mechanical strength make MWNTs the ultimate atomic force microscope probe tips. CNTs have the potential to be used as efficient molecular assembly devices for manufacturing nanomachines one atom at a time.

Two obvious nanotechnological applications of CNTs are nanobearings and nanosprings. Zettl and Cumings [53] have created MWNT-based linear bearings and constant force nanosprings. CNTs may potentially form the ultimate set of nanometer-sized building blocks, out of which nanomachines of all kinds can be built. These nanomachines can be used in the assembly of nanomachines, which can then be used to construct machines of all types and sizes. These machines can be competitive with, or perhaps surpass existing devices of all kinds.

SWNTs can also be used as electromechanical actuators. Baughman et al. [54] have demonstrated that sheets of SWNTs generate larger forces than natural muscle and larger strains than high-modulus ferroelectrics. They have predicted that actuators using optimized SWNT sheets may provide substantially higher work densities per cycle than any other known actuator. Kim and Lieber [55] have built SWNT and MWNT nanotweezers. These nanoscale electromechanical devices were used to manipulate and interrogate nanostructures. Electrically conducting CNTs were attached to electrodes on pulled glass micropipettes. Voltages applied to the electrodes opened and closed the free ends of the CNTs. Kim and Lieber demonstrated the capability of the nanotweezers by grabbing and manipulating submicron clusters and nanowires. This device could be used to manipulate biological cells or even manipulate organelles and clusters within human cells. Perhaps, more importantly, these tweezers can potentially be used to assemble other nanomachines.

A wide variety of nanoscale manipulators have been proposed [56] including pneumatic manipulators that can be configured to make tentacle, snake, or multi-chambered devices. Drexler has proposed telescoping nanomanipulators for precision molecular positioning and assembly work. His manipulator has a cylindrical shape with a diameter of 35 nm and an extensible length of 100 nm. A number of six

degree of freedom Stewart platforms have been proposed [56], including one that allows strut lengths to be moved in 0.10 nm increments across a 100 nm work envelope. A number of other nanodevices including box-spring accelerometers, displacement accelerometers, pivoted gyroscopic accelerometers, and gimbaled nanogyroscopes have been proposed and designed [56].

Currently, much thought is being devoted to molecular assembly and self-replicating devices (self-replicating nanorobots). Self-assembly is arguably the only way for nanotechnology to advance in an engineering or technological sense. Assembling a billion or trillion atom device—one atom at a time—would be a great accomplishment. It would take a huge investment in equipment, labor, and time. Freitas [56] describes the infrastructure needed to construct a simple medical nanorobot: a 1-$\mu$m spherical respirocyte consisting of about 18 billion atoms. He estimates that a factory production line deploying a coordinated system of 100 macroscale scanning probe microscope (SPM) assemblers, where each assembler is capable of depositing one atom per second on a convergently-assembled workpiece, would result in a manufacturing throughput of two nanorobots per decade. If one conjectures about enormous increases in assembler manufacturing rates even to the extent of an output of one nanorobot per minute, it would take two million years to build the first cubic centimeter therapeutic dosage of nanorobots. Thus, it is clear that the future of medical nanotechnology and nanoengineering lies in the direction of self-assembly and self-replication.

## References


1. Bridgman, P. W., *Dimensional Analysis,* 2nd Ed., Yale University Press, 1931.
2. Buckingham, E., "On physically similar systems: illustrations of the use of dimensional equations," *Physical Review,* 4(4):345–376, 1914.
3. Huntley, H. E., *Dimensional Analysis,* Dover Publications, 1967.
4. Langhaar, H. L., *Dimensional Analysis and Theory of Models,* John Wiley and Sons, 1951.
5. Taylor, E. S., *Dimensional Analysis for Engineers,* Oxford University Press, 1974.
6. Israelachvili, J. N., *Intermolecular and Surface Forces,* Academic Press, 1985, pp. 9–10.
7. Fearing, R. S., "Microactuators for microrobots: electric and magnetic," *Workshop on Micromechatronics, IEEE International Conference on Robotics and Automation,* 1997.
8. Bobbio, S. M., Keelam, M. D., Dudley, B. W., Goodwin-Hohansson, S., Jones, S. K., Jacobson, J. D., Tranjan, F. M., Dubois, T. D., "Integrated force arrays," *Proceedings of the IEEE Micro Electro Mechanical Systems,* 149–154, 1993.
9. Jacobson, J. D., Goodwin-Johansson, S. H., Bobbio, S. M., Bartlett, C. A., Yadon, L. N., "Integrated force arrays: theory and modeling of static operation," *Journal of Microelectromechanical Systems,* 4(3):139–150, 1995.
10. Yamaguchi, M., Kawamura, S., Minami, K., Esashi, M., "Distributed electrostatic micro actuators," *Proceedings of the IEEE Micro Electro Mechanical Systems,* 18–23, 1993.
11. Kim, C. J., Pisano, A. P., Muller, R. S., "Silicon-processed overhanging microgripper," *Journal of Microelectromechanical Systems,* 1(1):31–36, 1992.
12. Matsubara, T., Yamaguchi, M., Minami, K., Esashi, M., "Stepping electrostatic microactuator," *International Conference on Solid-State Sensor and Actuators,* 50–53, 1991.
13. Niino, T., Egawa, S., Kimura, H., Higuchi, T., "Electrostatic artificial muscle: compact, high-power linear actuators with multiple-layer structures," *Proceedings of the IEEE Conference on Micro Electro Mechanical Systems,* 130–135, 1994.
14. Huang, J. B., Mao, P. S., Tong, Q. Y., Zhang, R. Q., "Study on silicon electrostatic and electroquasistatic micromotors," *Sensors and Actuators,* 35:171–174, 1993.
15. Mehragany, M., Bart, S. F., Tavrow, L. S., Lang, J. H., Senturia, S. D., Schlecht, M. F., "A study of three microfabricated variable-capacitance motors," *Sensors and Actuators,* 173–179, 1990.
16. Trimmer, W., Gabriel, K., "Design considerations for a practical electrostatic micromotor," *Sensors and Actuators,* 11:189–206, 1987.

17. Mehregany, M., Nagarkar, P., Senturia, S. D., Lang, J. H., "Operation of microfabricated harmonic and ordinary side-drive motors," *Proceeding of the IEEE Conference on Micro Electro Mechanical Systems,* 1–8, 1990.
18. Dhuler, V. R., Mehregany, M., Phillips, S. M., "A comparative study of bearing designs and operational environments for harmonic side-drive micromotors," *IEEE Transactions on Electron Devices,* 40(11):1985–1989, 1993.
19. Price, R. H., Wood, J. E., Jacobsen, S. C., "Modeling considerations for electrostatic forces in electrostatic microactuators," *Sensors and Actuators,* 20:107–114, 1989.
20. Trimmer, W., Jebens, R., "An operational harmonic electrostatic motor," *Proceeding of the IEEE Conference on Micro Electro Mechanical Systems,* 13–16, 1989.
21. Trimmer, W., Jebens, R., "Harmonic electrostatic motors," *Sensors and Actuators,* 20:17–24, 1989.
22. Furuhata, T., Hirano, T., Lane, L. H., Fontanta, R. E., Fan, L. S., Fujita, H., "Outer rotor surface micromachined wobble micromotor," *Proceeding of the IEEE Conference on Micro Electro Mechanical Systems,* 161–166, 1993.
23. Trimmer, W., Jebens, R., "Actuators for microrobots," *IEEE Conference on Robotics and Automation,* 1547–1552, 1989.
24. Trimmer, W., "Microrobots and micromechanical systems," *Sensors and Actuators,* 19:267–287, 1989.
25. Inoue, T., Hamasaki, Y., Shimoyama, I., Miura, H., "Micromanipulation using a microcoil array," *Proceedings of the IEEE International Conference on Robotics and Automation,* 2208–2213, 1996.
26. Liu, C., Tsao, T., Tai, Y., Ho, C., "Surface micromachined magnetic actuators," *Proceedings of the IEEE Conference on Micro Electro Mechanical Systems,* 57–62, 1994.
27. Shinozawa, Y., Abe, T., Kondo, T., "A proportional microvalve using a bi-stable magnetic actuator," *Proceedings of the IEEE Conference on Micro Electro Mechanical Systems,* 233–237, 1997.
28. Wagner, B., Benecke, W., "Microfabricated actuator with moving permanent magnet," *Proceedings of the IEEE Conference on Micro Electro Mechanical Systems,* 27–32, 1991.
29. Yanagisawa, K., Tago, A., Ohkubo, T., Kuwano, H., "Magnetic microactuator," *Proceedings of the IEEE Conference on Micro Electro Mechanical Systems,* 120–124, 1991.
30. Judy, J., Muller, R. S., Zappe, H. H., "Magnetic microactuation of polysilicon flexure structures," *Journal of Microelectromechanical Systems,* 4(4):162–169, 1995.
31. Ahn, C. H., Kim, Y. J., Allen, M. G., "A planar variable reluctance magnetic micromotor with fully integrated stator and wrapped coils," *Proceedings of the IEEE Conference on Micro Electro Mechanical Systems,* 1–6, 1993.
32. Guckel, H., Christenson, T. R., Skrobis, K. J., Jung, T. S., Klein, J., Hartojo, K. V., Widjaja, I., "A first functional current excited planar rotational magnetic micromotor," *Proceedings of the IEEE Conference on Micro Electro Mechanical Systems,* 7–11, 1993.
33. Guckel, H., Sneigowski, J. J., Christenson, T. R., Raissi, F., "The application of fine grained, tensile polysilicon to mechanically resonant transducers," *Sensor and Actuators,* A21–A23:346–351, 1990.
34. Ikeda, K., Kuwayama, H., Kobayashi, T., Watanabe, T., Nishikawa, T., Yoshida, T., Harada, K., "Silicon pressure sensor integrates resonant strain gauge on diaphragm," *Sensors and Actuators,* A21–A23:146–150, 1990.
35. Fujii, T., Gotoh, Y., Kuroyanagi, S., "Fabrication of microdiaphragm pressure sensor utilizing micromachining," *Sensors and Actuators,* A34:217–224, 1992.
36. Mallon, J., Pourahmadi, F., Petersen, K., Barth, P., Vermeulen, T., Bryzek, J., "Low-pressure sensors employing bossed diaphragms and precision etch-stopping," *Sensors and Actuators,* A21–23:89–95, 1990.
37. Ikeda, K., Kuwayama, H., Kobayashi, T., Watanabe, T., Nishikawa, T., Yoshida, T., Harada, K., "Three-dimensional micromachining of silicon pressure sensor integrating resonant strain gauge on diaphragm," *Sensors and Actuators,* A21–A23:1007–1009, 1990.
38. Nagata, T., Terabe, H., Kuwahara, S., Sakurai, S., Tabata, O., Sugiyama, S., Esashi, M., "Digital compensated capacitive pressure sensor using cmos technology for low-pressure measurements," *Sensors and Actuators,* A34:173–177, 1992.

39. Stemme, E., Stemme, G., "A balanced resonant pressure sensor," *Sensors and Actuators,* A21–A23: 336–341, 1990.
40. Boxenhorn, B., Greiff, P., "Monolithic silicon accelerometer," *Sensors and Actuators,* A21–A23:273–277, 1990.
41. Leuthold, H., Rudolf, F., "An ASIC for high-resolution capacitive microaccelerometers," *Sensors and Actuators,* A21–A23:278–281, 1990.
42. Seidel, H., Riedel, H., Kolbeck, R., Muck, G., Kupke, W., Koniger, M., "Capacitive silicon accelerometer with highly symmetrical design," *Sensors and Actuators,* A21–A23:312–315, 1990.
43. Chang, S. C., Putty, M. W., Hicks, D. B., Li, C. H., Howe, R. T., "Resonant-bridge two-axis microaccelerometer," *Sensors and Actuators,* A21–A23:342–345, 1990.
44. Satchell, D. W., Greenwood, J. C., "A thermally-excited silicon accelerometer," *Sensors and Actuators,* A17:241–245, 1989.
45. Chau, K. H. L., Lewis, S. R., Zhao, Y., Howe, R. T., Bart, S. F., Marchesilli, R. G., "An integrated force-balanced capacitive accelerometer for low-*g* applications," *Sensors and Actuators,* A54:472–476, 1996.
46. Kuehnel, W., Sherman, S., "A surface micromachined silicon accelerometer with on-chip detection circuitry," *Sensors and Actuators,* A45:7–16, 1994.
47. Despont, Racine, G. A., Renaud, P., de Rooij, N. F., "New design of micromachined capacitive force sensor," *Journal of Micromechanics and Microengineering,* 3:239–242, 1993.
48. Blom, F. R., Bouwstra, S., Fluitman, J. H. J., Elwenspoek, M., "Resonating silicon beam force sensor," *Sensors and Actuators,* 17:513–519, 1989.
49. Bernstein, J., Cho, S., King, A. T., Kourepenis, A., Maciel, P., Weinberg, M., "A micromachined comb-drive tuning fork rate gyroscope," *IEEE Conference on Micro Electro Mechanical Systems,* 143–148, 1993.
50. Oh, Y., Lee, B., Baek, S., Kim, H., Kim, J., Kang, S., Song, C., "A surface-micromachined tunable vibratory gyroscope," *IEEE Conference on Micro Electro Mechanical Systems,* 272–277, 1997.
51. Venturi, M., Credi, A., Balzani, V., "Devices and machines at the molecular level," *Electronic Properties of Novel Materials, AIP Conf. Proc.,* 544:489–494, 2000.
52. Ajayan, P. M., Charlier, J. C., Rinzler, A. G., "PNAS," 96:14199–14200, 1999.
53. Zettl, A., Cumings, J., "Sharpened nanotubes, nanobearings and nanosprings," *Electronic Properties of Novel Materials, AIP Conf. Proc.,* 544:526–531, 2000.
54. Baughman, R. H., et al., "Carbon nanotube actuators," *Science,* 284:1340–1344, 1999.
55. Kim, P., Lieber, C. M., "Nanotube nanotweezers," *Science,* 286:2148–2150, 1999.
56. Freitas, R. A., "Nanomedicine," Vol. 1, *Landes Bioscience,* Austin, 1999.

# 6

# Modeling Electromechanical Systems




Francis C. Moon
*Cornell University*


## 6.1 Introduction

Mechatronics describes the integration of mechanical, electromagnetic, and computer elements to produce devices and systems that monitor and control machine and structural systems. Examples include familiar consumer machines such as VCRs, automatic cameras, automobile air bags, and cruise control devices. A distinguishing feature of modern mechatronic devices compared to earlier controlled machines is the miniaturization of electronic information processing equipment. Increasingly computer and electronic sensors and actuators can be embedded in the structures and machines. This has led to the need for integration of mechanical and electrical design. This is true not only for sensing and signal processing but also for actuator design. In human size devices, more powerful magnetic materials and superconductors have led to the replacement of hydraulic and pneumatic actuators with servo motors, linear motors, and other electromagnetic actuators. At the material scale and in microelectromechanical systems (MEMS), electric charge force actuators, piezoelectric actuators, and ferroelectric actuators have made great strides.

While the materials used in electromechanical design are often new, the basic dynamic principles of Newton and Maxwell still apply. In spatially extended systems one must solve continuum problems using the theory of elasticity and the partial differential equations of electromagnetic field theory. For many applications, however, it is sufficient to use lumped parameter modeling based on i) rigid body dynamics

for inertial components, ii) Kirchhoff circuit laws for current-charge components, and iii) magnet circuit laws for magnetic flux devices.

In this chapter we will examine the basic modeling assumptions for inertial, electric, and magnetic circuits, which are typical of mechatronic systems, and will summarize the dynamic principles and interactions between the mechanical motion, circuit, and magnetic state variables. We will also illustrate these principles with a few examples as well as provide some bibliography to more advanced references in electromechanics.

## 6.2 Models for Electromechanical Systems

The fundamental equations of motion for physical continua are partial differential equations (PDEs), which describe dynamic behavior in both time and space. For example, the motions of strings, elastic beams and plates, fluid flow around and through bodies, as well as magnetic and electric fields require both spatial and temporal information. These equations include those of elasticity, elastodynamics, the Navier–Stokes equations of fluid mechanics, and the Maxwell–Faraday equations of electromagnetics. Electromagnetic field problems may be found in Jackson (1968). Coupled field problems in electric fields and fluids may be found in Melcher (1980) and problems in magnetic fields and elastic structures may be found in the monograph by Moon (1984). This short article will only treat solid systems.

Many practical electromechanical devices can be modeled by lumped physical elements such as mass or inductance. The equations of motion are then integral forms of the basic PDEs and result in coupled ordinary differential equations (ODEs). This methodology will be explored in this chapter. Where physical problems have spatial distributions, one can often separate the problem into spatial and temporal parts called *separation of variables*. The spatial description is represented by a finite number of spatial or eigenmodes each of which has its modal amplitude. This method again results in a set of ODEs. Often these coupled equations can be understood in the context of simple lumped mechanical masses and electric and magnetic circuits.

## 6.3 Rigid Body Models

### Kinematics of Rigid Bodies

Kinematics is the description of motion in terms of position vectors **r**, velocities **v**, acceleration **a**, rotation rate vector $\omega$, and generalized coordinates $\{q_k(t)\}$ such as relative angular positions of one part to another in a machine (Figure 6.1). In a rigid body one generally specifies the position vector of one point, such as the center of mass $\mathbf{r}_c$, and the velocity of that point, say $\mathbf{v}_c$. The angular position of a rigid body is specified by angle sets call Euler angles. For example, in vehicles there are pitch, roll, and yaw angles (see, e.g., Moon, 1999). The angular velocity vector of a rigid body is denoted by $\omega$. The velocity of a point in a rigid body other than the center of mass, $\mathbf{r}_p = \mathbf{r}_c + \rho$, is given by

$$\mathbf{v}_P = \mathbf{v}_c + \omega \times \rho \tag{6.1}$$

where the second term is a vector cross product. The angular velocity vector $\omega$ is a property of the entire rigid body. In general a rigid body, such as a satellite, has six degrees of freedom. But when machine elements are modeled as a rigid body, kinematic constraints often limit the number of degrees of freedom.

### Constraints and Generalized Coordinates

Machines are often collections of rigid body elements in which each component is constrained to have one degree of freedom relative to each of its neighbors. For example, in a multi-link robot arm shown in Figure 6.2, each rigid link has a revolute degree of freedom. The degrees of freedom of each rigid link are constrained by bearings, guides, and gearing to have one type of relative motion. Thus, it is convenient

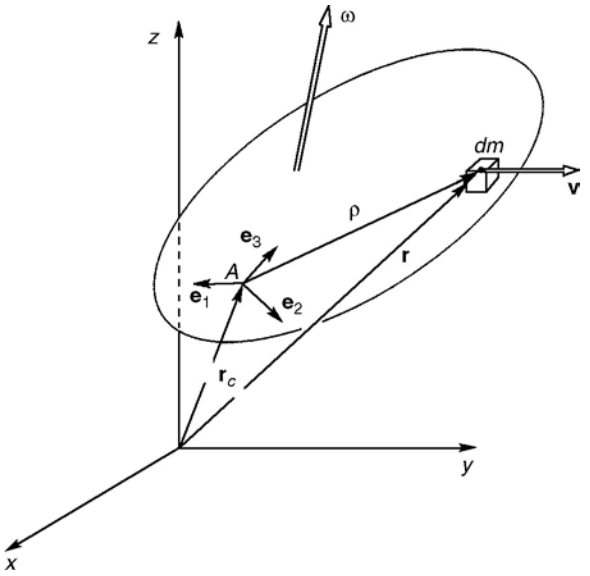



**FIGURE 6.1** Sketch of a rigid body with position vector, velocity, and angular velocity vectors.

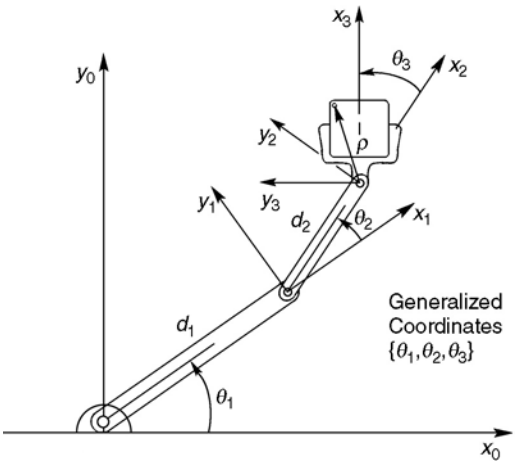



**FIGURE 6.2** Multiple link robot manipulator arm.

to use these generalized motions $\{q_k\colon k = 1,\ldots,K\}$ to describe the dynamics. It is sometimes useful to define a vector or matrix, $\mathbf{J}(q_k)$, called a *Jacobian*, that relates velocities of physical points in the machine to the generalized velocities $\{\dot{q}_k\}$. If the position vector to some point in the machine is $\mathbf{r}_P(q_k)$ and is determined by geometric constraints indicated by the functional dependence on the $\{q_k(t)\}$, then the velocity of that point is given by

$$\mathbf{v}_P = \sum \frac{\partial \mathbf{r}_P}{\partial q_r}\dot{q}_r = \mathbf{J}\cdot\dot{\mathbf{q}} \tag{6.2}$$

where the sum is on the number of generalized degrees of freedom $K$. The three-by-$K$ matrix $\mathbf{J}$ is called a *Jacobian* and $\dot{\mathbf{q}}$ is a $K \times 1$ vector of generalized coordinates. This expression can be used to calculate the kinetic energy of the constrained machine elements, and using Lagrange's equations discussed below, derive the equations of motion (see also Moon, 1999).

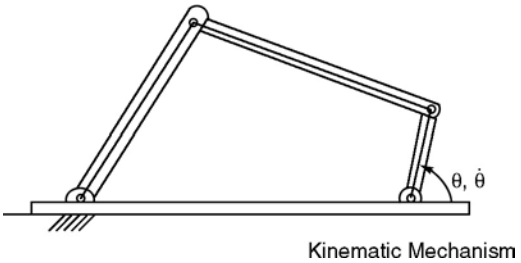



**FIGURE 6.3** Example of a kinematic mechanism.

### Kinematic versus Dynamic Problems

Some machines are constructed in a closed kinematic chain so that the motion of one link determines the motion of the rest of the rigid bodies in the chain, as in the four-bar linkage shown in Figure 6.3. In these problems the designer does not have to solve differential equations of motion. Newton's laws are used to determine forces in the machine, but the motions are *kinematic*, determined through the geometric constraints.

In open link problems, such as robotic devices (Figure 6.2), the motion of one link does not determine the dynamics of the rest. The motions of these devices are inherently *dynamic*. The engineer must use both the kinematic constraints (6.2) as well as the Newton–Euler differential equation of motion or equivalent forms such as Lagrange's equation discussed below.

## 6.4 Basic Equations of Dynamics of Rigid Bodies

In this section we review the equations of motion for the mechanical plant in a mechatronics system. This plant could be a system of rigid bodies such as in a serial robot manipulator arm (Figure 6.2) or a magnetically levitated vehicle (Figure 6.4), or flexible structures in a MEMS accelerometer. The dynamics of flexible structural systems are described by PDEs of motion. The equation for rigid bodies involves Newton's law for the motion of the center of mass and Euler's extension of Newton's laws to the angular momentum of the rigid body. These equations can be formulated in many ways (see Moon, 1999):

1. Newton–Euler equation (vector method)
2. Lagrange's equation (scalar-energy method)
3. D'Alembert's principle (virtual work method)
4. Virtual power principle (Kane's equation, or Jourdan's principle)

### Newton–Euler Equation

Consider the rigid body in Figure 6.1 whose center of mass is measured by the vector $\mathbf{r}_c$ in some fixed coordinate system. The velocity and acceleration of the center of mass are given by

$$\dot{\mathbf{r}}_c = \mathbf{v}_c, \qquad \dot{\mathbf{v}}_c = \mathbf{a}_c \tag{6.3}$$

The "over dot" represents a total derivative with respect to time. We represent the total sum of vector forces on the body from both mechanical and electromagnetic sources by **F**. Newton's law for the motion of the center of mass of a body with mass $m$ is given by

$$m\dot{\mathbf{v}}_c = \mathbf{F} \tag{6.4}$$

**FIGURE 6.4** Magnetically levitated rigid body (HSST MagLev prototype vehicle, 1998, Nagoya, Japan).

If $\mathbf{r}$ is a vector to some point in the rigid body, we define a local position vector $\rho$ by $\mathbf{r}_P = \mathbf{r}_c + \rho$. If a force $\mathbf{F}_i$ acts at a point $\mathbf{r}_i$ in a rigid body, then we define the moment of the force $\mathbf{M}$ about the fixed origin by

$$\mathbf{M}_i = \mathbf{r}_i \times \mathbf{F}_i \tag{6.5}$$

The total force moment is then given by the sum over all the applied forces as the body

$$\mathbf{M} = \sum \mathbf{r}_i \times \mathbf{F}_i = \mathbf{r}_c \times \mathbf{F} + \mathbf{M}_c \quad \text{where} \quad \mathbf{M}_c = \sum \rho_i \times \mathbf{F}_i \tag{6.6}$$

We also define the *angular momentum* of the rigid body by the product of a symmetric matrix of second moments of mass called the *inertia matrix* $\mathbf{I}_c$. The angular momentum vector about the center of mass is defined by

$$\mathbf{H}_c = \mathbf{I}_c \cdot \omega \tag{6.7}$$

Since $\mathbf{I}_c$ is a symmetric matrix, it can be diagonalized with principal inertias (or eigenvalues) $\{I_{ic}\}$ about principal directions (eigenvectors) $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$. In these coordinates, which are attached to the body, the angular momentum about the center of mass becomes

$$\mathbf{H}_c = I_{1c}\omega_1\mathbf{e}_1 + I_{2c}\omega_2\mathbf{e}_2 + I_{3c}\omega_3\mathbf{e}_3 \tag{6.8}$$

where the angular velocity vector is written in terms of principal eigenvectors $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ attached to the rigid body.

Euler's extension of Newton's law for a rigid body is then given by

$$\dot{\mathbf{H}}_c = \mathbf{M}_c \tag{6.9}$$

This equation says that the change in the angular momentum about the center of mass is equal to the total moment of all the forces about the center of mass. The equation can also be applied about a fixed point of rotation, which is not necessarily the center of mass, as in the example of the compound pendulum given below.

Equations (6.4) and (6.9) are known as the Newton–Euler equations of motion. Without constraints, they represent six coupled second order differential equations for the position of the center of mass and for the angular orientation of the rigid body.

### Multibody Dynamics

In a serial link robot arm, as shown in Figure 6.2, we have a set of connected rigid bodies. Each body is subject to both applied and constraint forces and moments. The dynamical equations of motion involve the solution of the Newton–Euler equations for each rigid link subject to the geometric or kinematics constraints between each of the bodies as in (6.2). The forces on each body will have applied terms $\mathbf{F}^a$, from actuators or external mechanical sources, and internal constraint forces $\mathbf{F}^c$. When friction is absent, the work done by these constraint forces is zero. This property can be used to write equations of motion in terms of scalar energy functions, known as Lagrange's equations (see below).

Whatever the method used to derive the equation of motions, the dynamical equations of motion for multibody systems in terms of generalized coordinates $\{q_k(t)\}$ have the form

$$\sum m_{ij}\ddot{q}_j + \sum\sum \mu_{ijk}\dot{q}_j\dot{q}_k = Q_i \tag{6.10}$$

The first term on the left involves a generalized symmetric mass matrix $m_{ij} = m_{ji}$. The second term includes Coriolis and centripetal acceleration. The right-hand side includes all the force and control terms. This equation has a quadratic nonlinearity in the generalized velocities. These quadratic terms usually drop out for rigid body problems with a single axis of rotation. However, the nonlinear inertia terms generally appear in problems with simultaneous rotation about two or three axes as in multi-link robot arms (Figure 6.2), gyroscope problems, and slewing momentum wheels in satellites.

In modern dynamic simulation software, called multibody codes, these equations are automatically derived and integrated once the user specifies the geometry, forces, and controls. Some of these codes are called ADAMS, DADS, Working Model, and NEWEUL. However, the designer must use caution as these codes are sometimes poor at modeling friction and impacts between bodies.

## 6.5 Simple Dynamic Models

Two simple examples of the application of the angular momentum law are now given. The first is for rigid body rotation about a single axis and the second has two axes of rotation.

### Compound Pendulum

When a body is constrained to a single rotary degree of freedom and is acted on by the force of gravity as in Figure 6.5, the equation of motion takes the form, where $\theta$ is the angle from the vertical,

$$I\ddot{\vartheta} - (m_1L_1 - m_2L_2)g \sin\theta = T(t) \tag{6.11}$$

where $T(t)$ is the applied torque, $I = m_1L_1^2 + m_2L_2^2$ is the moment of inertia (properly called the second moment of mass). The above equation is nonlinear in the sine function of the angle. In the case of small motions about $\theta = 0$, the equation becomes a linear differential equation and one can look for solutions of the form $\theta = A\cos\omega t$, when $T(t) = 0$. For this case the pendulum exhibits sinusoidal motion with natural frequency

$$\omega = [g(m_2L_2 - m_1L_1)/I]^{1/2} \tag{6.12}$$

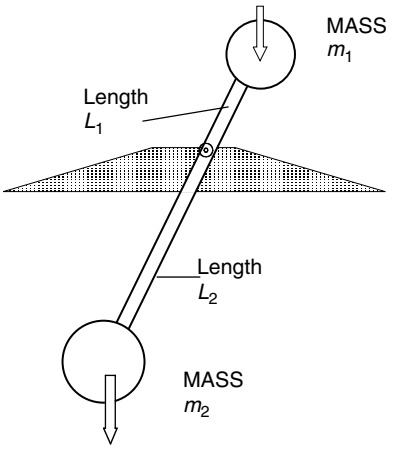



**FIGURE 6.5** Sketch of a compound pendulum under gravity torques.

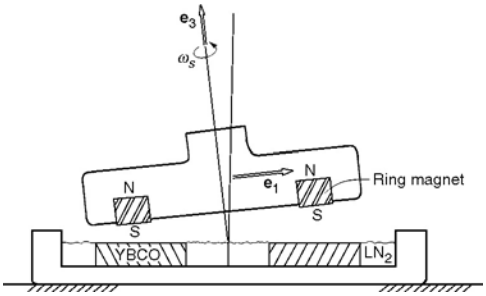



**FIGURE 6.6** Sketch of a magnetically levitated flywheel on high-temperature superconducting bearings.

For the simple pendulum $m_1 = 0$, and we have the classic pendulum relation in which the natural frequency depends inversely on the square root of the length:

$$\omega = (g/L_2)^{1/2} \tag{6.13}$$

## Gyroscopic Motions

Spinning devices such as high speed motors in robot arms or turbines in aircraft engines or magnetically levitated flywheels (Figure 6.6) carry angular momentum, devoted by the vector **H**. Euler's extension of Newton's laws says that a change in angular momentum must be accompanied by a force moment **M**,

$$\mathbf{M} = \dot{\mathbf{H}} \tag{6.14}$$

In three-dimensional problems one can often have components of angular momentum about two different axes. This leads to a Coriolis acceleration that produces a gyroscopic moment even when the two angular motions are steady. Consider the spinning motor with spin $\dot{\phi}$ about an axis with unit vector $\mathbf{e}_1$ and let us imagine an angular motion of the $\mathbf{e}_1$ axis, $\dot{\psi}$ about a perpendicular axis $\mathbf{e}_z$ called the precession axis in gyroscope parlance. Then one can show that the angular momentum is given by

$$\mathbf{H} = I_1\dot{\phi}\mathbf{e}_1 + I_z\dot{\psi}\mathbf{e}_z \tag{6.15}$$

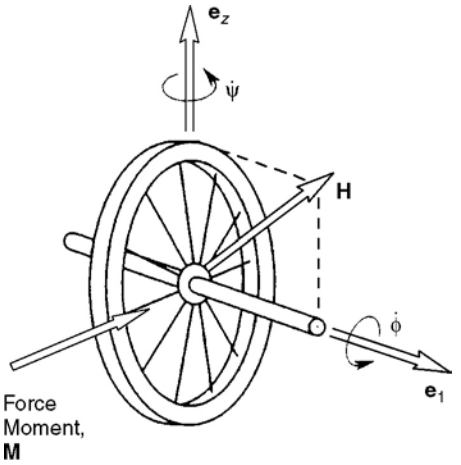



**FIGURE 6.7** Gyroscopic moment on a precessing, spinning rigid body.

and the rate of change of angular momentum for constant spin and presession rates is given by

$$\dot{\mathbf{H}} = \dot{\psi}\mathbf{e}_z \times \mathbf{H} \tag{6.16}$$

There must then exist a gyroscopic moment, often produced by forces on the bearings of the axel (Figure 6.7). This moment is perpendicular to the plane formed by $\mathbf{e}_1$ and $\mathbf{e}_z$, and is proportional to the product of the rotation rates:

$$\mathbf{M} = I_1 \dot{\phi}\dot{\psi}\mathbf{e}_z \times \mathbf{e}_1 \tag{6.17}$$

This has the same form as Equation (6.10), when the generalized force $Q$ is identified with the moment $\mathbf{M}$, i.e., the moment is the product of generalized velocities when the second derivative acceleration terms are zero.

## 6.6 Elastic System Modeling

Elastic structures take the form of cables, beams, plates, shells, and frames. For linear problems one can use the method of eigenmodes to represent the dynamics with a finite set of modal amplitudes for generalized degrees of freedom. These eigenmodes are found as solutions to the PDEs of the elastic structure (see, e.g., Yu, 1996).

The simplest elastic structure after the cable is a one-dimensional beam shown in Figure 6.8. For small motions we assume only transverse displacements $w(x, t)$, where $x$ is a spatial coordinate along the beam. One usually assumes that the stresses on the beam cross section can be integrated to obtain stress vector resultants of shear $V$, bending moment $M$, and axial load $T$. The beam can be loaded with point or concentrated forces, end forces or moment or distributed forces as in the case of gravity, fluid forces, or electromagnetic forces. For a distributed transverse load $f(x, t)$, the equation of motion is given by

$$D\frac{\partial^4 w}{\partial x^4} - T\frac{\partial^2 w}{\partial x^2} + \rho A\frac{\partial^2 w}{\partial t^2} = f(x, t) \tag{6.18}$$

where $D$ is the bending stiffness, $A$ is the cross-sectional area of the beam, and $\rho$ is the density. For a beam with Young's modulus $Y$, rectangular cross section of width $b$, and height $h$, $D = Ybh^3/12$. For $D = 0$, one has a cable or string under tension $T$, and the equation takes the form of the usual wave equation. For a beam with tension $T$, the natural frequencies are increased by the addition of the second term in the

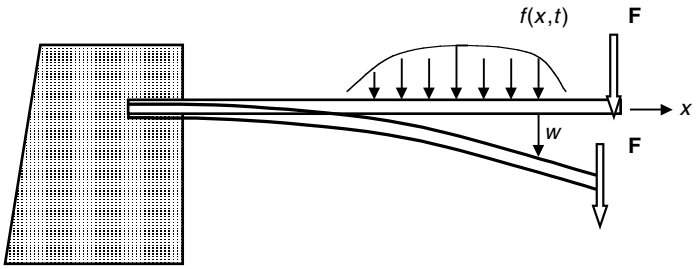



**FIGURE 6.8** Sketch of an elastic cantilevered beam.

equation. For $T = -P$, i.e., a compressive load on the end of the beam, the curvature term leads to a decrease of natural frequency with increase of the compressive force $P$. If the lowest natural frequency goes to zero with increasing load $P$, the straight configuration of the beam becomes unstable or undergoes *buckling*. The use of $T$ or $(-P)$ to stiffen or destiffen a beam structure can be used in design of sensors to create a sensor with variable resonance. This idea has been used in a MEMS accelerometer design (see below).

Another feature of the beam structure dynamics is the fact that unlike the string or cable, the frequencies of the natural modes are not commensurate due to the presence of the fourth-order derivative term in the equation. In wave type problems this is known as *wave dispersion*. This means that waves of different wavelengths travel at different speeds so that wave pulse shapes change their form as the wave moves through the structure.

In order to solve dynamic problems in finite length beam structures, one must specify boundary conditions at the ends. Examples of boundary conditions include

$$\begin{aligned} &\text{clamped end} && w = 0, \quad \frac{\partial w}{\partial x} = 0 \\ &\text{pinned end} && w = 0, \quad \frac{\partial^2 w}{\partial x^2} = 0 \text{ (zero moment)} \\ &\text{free end} && \frac{\partial^2 w}{\partial x^2} = 0, \quad \frac{\partial^3 w}{\partial x^3} = 0 \text{ (zero shear)} \end{aligned} \tag{6.19}$$

## Piezoelastic Beam

Piezoelastic materials exhibit a coupling between strain and electric polarization or voltage. Thus, these materials can be used for sensors or actuators. They have been used for active vibration suppression in elastic structures. They have also been explored for active optics space applications. Many natural materials exhibit piezoelasticity such as quartz as well as manufactured materials such as barium titanate, lead zirconate titanate (PZT), and polyvinylidene fluoride (PVDF). Unlike forces on charges and currents (see below), the electric effect takes place through a change in shape of the material. The modeling of these devices can be done by modifying the equations for elastic structures.

The following work on piezo-benders is based on the work of Lee and Moon (1989) as summarized in Miu (1993). One of the popular configurations of a piezo actuator-sensor is the piezo-bender shown in Figure 6.9. The elastic beam is of rectangular cross section as is the piezo element. The piezo element can be cemented on one or both sides of the beam either partially or totally covering the surface of the non-piezo substructure.

In general the local electric dipole polarization depends on the six independent strain components produced by normal and shear stresses. However, we will assume that the transverse voltage or polarization is coupled to the axial strain in the plate-shaped piezo layers. The constitutive relations between

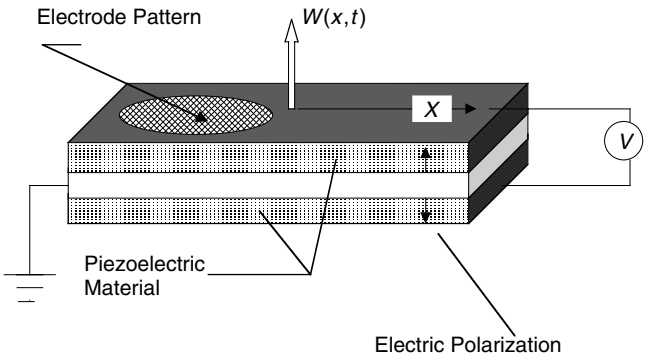



**FIGURE 6.9** Elastic beam with two piezoelectric layers (Lee and Moon, 1989).

axial stress and strain, $T$, $S$, electric field and electric displacement, $E_3$, $D_3$ (not to be confused with the bending stiffness $D$), are given by

$$T_1 = c_{11}S_1 - e_{31}E_3, \qquad D_3 = e_{31}S_1 + \varepsilon_3 E_3 \tag{6.20}$$

The constants $c_{11}$, $e_{31}$, $\varepsilon_3$, are the elastic stiffness modulus, piezoelectric coupling constant, and the electric permittivity, respectively.

If the piezo layers are polled in the opposite directions, as shown in the Figure 6.9, an applied voltage will produce a strain extention in one layer and a strain contraction in the other layer, which has the effect of an applied moment on the beam. The electrodes applied to the top and bottom layers of the piezo layers can also be shaped so that there can be a gradient in the average voltage across the beam width. For this case the equation of motion of the composite beam can be written in the form

$$D\frac{\partial^4 w}{\partial x^4} + \rho A\frac{\partial^2 w}{\partial t^2} = -2e_{31}z_o\frac{\partial^2 V_3}{\partial x^2} \tag{6.21}$$

where $z_o = (h_S + h_P)/2$.

The $z$ term is the average of piezo plate and substructure thicknesses. When the voltage is uniform, then the right-hand term results in an applied moment at the end of the beam proportional to the transverse voltage.

## 6.7 Electromagnetic Forces

One of the keys to modeling mechatronic systems is the identification of the electric and magnetic forces. Electric forces act on charges and electric polarization (electric dipoles). Magnetic forces act on electric currents and magnetic polarization. Electric charge and current can experience a force in a uniform electric or magnetic field; however, electric and magnetic dipoles will only produce a force in an electric or magnetic field gradient.

Electric and magnetic forces can also be calculated using both direct vector methods as well as from energy principles. One of the more popular methods is *Lagrange's equation* for electromechanical systems described below.

Electromagnetic systems can be modeled as either distributed field quantities, such as electric field **E** or magnetic flux density **B** or as lumped element electric and magnetic circuits. The force on a point charge $Q$ is given by the vector equation (Figure 6.10):

$$\mathbf{F} = Q\mathbf{E} \tag{6.22}$$

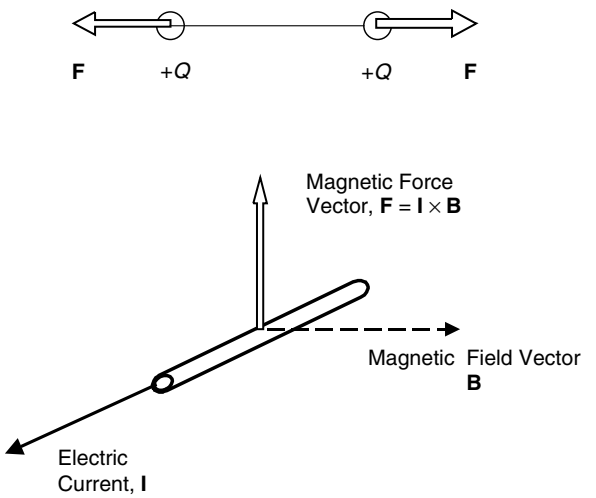



**FIGURE 6.10** Electric forces on two charges (top). Magnetic force on a current carrying wire element (bottom).

When **E** is generated by a single charge, the force between charges $Q_1$ and $Q_2$ is given by

$$F = \frac{Q_1 Q_2}{4\pi\varepsilon_0 r^2} \tag{6.23}$$

and is directed along the line connecting the two charges. Like charges repel and opposite charges attract one another.

The magnetic force per unit length on a current element **I** is given by the cross product

$$\mathbf{F} = \mathbf{I} \times \mathbf{B} \tag{6.24}$$

where the magnetic force is perpendicular to the plane of the current element and the magnetic field vector. The total force on a closed circuit in a uniform field can be shown to be zero. Net forces on closed circuits are produced by field gradients due to other current circuits or field sources.

Forces produced by field distributions around a volume containing electric charge or current can be calculated using the field quantities of **E**, **B** directly using the concept of magnetic and electric stresses, which was developed by Faraday and Maxwell. These electromagnetic stresses must be integrated over an area surrounding the charge or current distribution. For example, a solid containing a current distribution can experience a *magnetic pressure*, $P = B_t^2/2\mu_0$, on the surface element and a *magnetic tension*, $t_n = B_n^2/2\mu_0$, where the magnetic field components are written in terms of values tangential and normal to the surface. Thus, a one-tesla magnetic field outside of a solid will experience 40 N/cm$^2$ pressure if the field is tangential to the surface.

In general there are four principal methods to calculate electric and magnetic forces:

- direct force vectors and moments between electric charges, currents, and dipoles;
- electric field-charge and magnetic field-current force vectors;
- electromagnetic tensor, integration of electric tension, magnetic pressure over the surface of a material body; and
- energy methods based on gradients of magnetic and electric energy.

Examples of the direct method and stress tensor method are given below. The energy method is described in the section on Lagrange's equations.

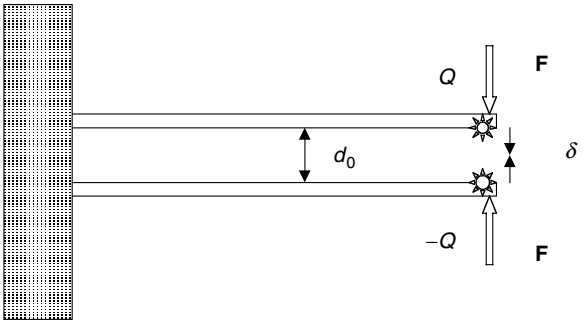



**FIGURE 6.11** Two elastic beams with electric charges at the ends.

### Example 1. Charge–Charge Forces

Suppose two elastic beams in a MEMS device have electric charges $Q_1$, $Q_2$ coulombs each concentrated at their tips (Figure 6.11). The electric force between the charges is given by the vector

$$\mathbf{F} = \frac{Q_1 Q_2}{4\pi\varepsilon_0}\frac{\mathbf{r}}{r^3} \quad \text{(newtons)} \tag{6.25}$$

where $1/4\pi\varepsilon_0 = 8.99 \times 10^9\ \text{Nm}^2/\text{C}^2$.

If the initial separation between the beams is $d_0$, we seek the new separation under the electric force. For simplicity, we let $Q_1 = -Q_2 = Q$, where opposite charges create an attractive force between the beam tips. The deflection of the cantilevers is given by

$$\delta = \frac{FL^3}{3YI} = \frac{1}{k}F \tag{6.26}$$

where $L$ is the length, $Y$ the Young's modulus, $I$ the second moment of area, and $k$ the effective spring constant.

Under the electric force, the new separation is $d = d_0 - 2\delta$,

$$k\delta = \frac{Q^2}{4\pi\varepsilon_0}\frac{1}{(d_0 - 2\delta)^2} \tag{6.27}$$

For $\delta \ll d_0$ to first order we have

$$\delta = \frac{Q^2/4\pi\varepsilon_0 d_0^2 k}{1 - (1/d_0^3)(Q^2/k\pi\varepsilon_0)} \tag{6.28}$$

This problem shows the potential for electric field buckling because as the beam tips move closer together, the attractive force between them increases. The nondimensional expression in the denominator

$$\frac{Q^2}{\pi\varepsilon_0 d_0^3}\frac{1}{k} \tag{6.29}$$

is the ratio of the negative electric stiffness to the elastic stiffness $k$ of the beams.

### Example 2. Magnetic Force on an Electromagnet

Imagine a ferromagnetic keeper on an elastic restraint of stiffness $k$, as shown in Figure 6.12. Under the soft magnetic keeper, we place an electromagnet which produces $N$ turns of current $I$ around a soft ferromagnetic core. The current is produced by a voltage in a circuit with resistance $R$.

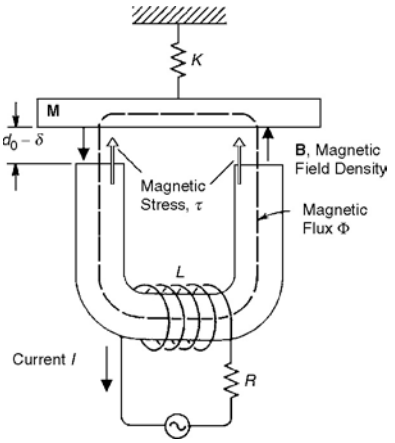



**FIGURE 6.12** Force on a ferromagnetic bar near an electromagnet.

The magnetic force will be calculated using the *magnetic stress tensor* developed by Maxwell and Faraday (see, e.g., Moon, 1984, 1994). Outside a ferromagnetic body, the stress tensor is given by **t** and the stress vector on the surface defined by normal **n** is given by $\boldsymbol{\tau} = \mathbf{t} \cdot \mathbf{n}$:

$$\tau = \frac{1}{\mu_0}\left(\frac{1}{2}[B_n^2 - B_t^2],\ B_n B_t\right) = (\tau_n, \tau_t) \tag{6.30}$$

For high magnetic permeability as in a ferromagnetic body, the tangential component of the magnetic field outside the surface is near zero. Thus the force is approximately normal to the surface and is found from the integral of the magnetic tension over the surface:

$$\mathbf{F} = \frac{1}{2\mu_0}\int B_n^2 \mathbf{n}\, dA \tag{6.31}$$

and $B_n^2/2\mu_0$ represents a magnetic tensile stress. Thus, if the area of the pole pieces of the electromagnet is $A$ (neglecting fringing of the field), the force is

$$F = B_g^2 A/\mu_0 \tag{6.32}$$

where $B_g$ is the gap field. The gap field is determined from Amperes law

$$NI = \widehat{R}\,\Phi, \quad \Phi = B_g A \tag{6.33}$$

where the *reluctance* is approximately given by

$$\widehat{R} = \frac{2(d_0 - \delta)}{\mu_0 A} \tag{6.34}$$

The balance of magnetic and elastic forces is then given by

$$F = \frac{1}{\mu_0 A}\Phi^2 = \frac{1}{\mu_0 A}\left(\frac{NI}{\widehat{R}}\right)^2 = k\delta \tag{6.35}$$

or

$$\frac{(NI)^2}{4(d_0-\delta)^2}\mu_0 A = k\delta, \qquad \frac{\mu_0 N^2 I^2 A}{4(d_0-\delta)^2} = k\delta$$

(Note that the expression $\mu_0 N^2 I^2$ has units of force.) Again as the current is increased, the total elastic and electric stiffness goes to zero and one has the potential for buckling.

## 6.8 Dynamic Principles for Electric and Magnetic Circuits

The fundamental equations of electromagnetics stem from the work of nineteenth century scientists such as Faraday, Henry, and Maxwell. They take the form of partial differential equations in terms of the field quantities of electric field **E** and magnetic flux density **B**, and also involve volumetric measures of charge density $q$ and current density **J** (see, e.g., Jackson, 1968). Most practical devices, however, can be modeled with lumped electric and magnetic circuits. The standard resistor, capacitor, inductor circuit shown in Figure 6.13 uses electric current $I$ (amperes), charge $Q$ (columbs), magnetic flux $\Phi$ (webers), and voltage $V$ (volts) as dynamic variables. The voltage is the integral of the electric field along a path:

$$V_{21} = \int_1^2 \mathbf{E}\cdot d\mathbf{l} \tag{6.36}$$

The charge $Q$ is the integral of charge density $q$ over a volume, and electric current $I$ is the integral of normal component of **J** across an area. The magnetic flux $\Phi$ is given as another surface integral of magnetic flux.

$$\Phi = \int \mathbf{B}\cdot d\mathbf{A} \tag{6.37}$$

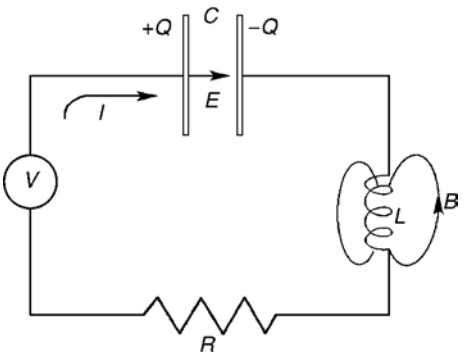



**FIGURE 6.13** Electric circuit with lumped parameter capacitance, inductance, and resistance.

When there are no mechanical elements in the system, the dynamical equations take the form of conservation of charge and the Faraday–Henry law of flux change.

$$\frac{dQ}{dt} = I \quad \text{(Conservation of charge)} \tag{6.38}$$

$$\frac{d\phi}{dt} = V \quad \text{(Law of flux change)} \tag{6.39}$$

where $\phi = N\Phi$ is called the number of flux linkages, and $N$ is an integer. In electromagnetic circuits the analog of mechanical constitutive properties is inductance $L$ and capacitance $C$. The magnetic flux in an inductor, for example, often depends on the current $I$.

$$\phi = f(I) \tag{6.40}$$

For a linear inductor we have a definition of inductance $L$, i.e., $\phi = LI$. If the system has a mechanical state variable such as displacement $x$, as in a magnetic solenoid actuator, then $L$ may be a function of $x$.

In charge storage circuit elements, the capacitance $C$ is defined as

$$Q = CV \tag{6.41}$$

In MEMS devices and in microphones, the capacitance may also be a function of some generalized mechanical displacement variable.

The voltages across the different circuit elements can be active or passive. A pure voltage source can maintain a given voltage, but the current depends on the passive voltages across the different circuit elements as summarized in the Kirchhoff circuit law:

$$\frac{d}{dt}L(x)I + \frac{Q}{C(x)} + RI = V(t) \tag{6.42}$$

## Lagrange's Equations of Motion for Electromechanical Systems

It is well known that the Newton–Euler equations of motion for mechanical systems can be derived using an energy principle called Lagrange's equation. In this method one identifies generalized coordinates $\{q_k\}$, not to be confused with electric charges, and writes the kinetic energy of the system $T$ in terms of generalized velocities and coordinates, $T(\dot{q}_k, q_k)$. Next the mechanical forces are split into so-called conservative forces, which can be derived from a potential energy function $W(q_k)$ and the rest of the forces, which are represented by a generalized force $Q_k$ corresponding to the work done by the $k$th generalized coordinate. Lagrange's equations for mechanical systems then take the form:

$$\frac{d}{dt}\frac{\partial T(\dot{q}_k, q_k)}{\partial \dot{q}_k} - \frac{\partial T}{\partial q_k} + \frac{\partial W(q_k)}{\partial q_k} = Q_k \tag{6.43}$$

For example, in a linear spring–mass–damper system, with mass $m$, spring constant $k$, viscous damping constant $c$, and one generalized coordinate $q_1 = x$, the equation of motion can be derived using, $T = \frac{1}{2}m\dot{x}^2$, $W = \frac{1}{2}kx^2$, $Q_1 = -c\dot{x}$, in Lagrange's equation above. What is remarkable about this formulation is that it can be extended to treat both electromagnetic circuits and coupled electromechanical problems.

As an example of the application of Lagrange's equations to a coupled electromechanical problem, consider the one-dimensional mechanical device, shown in Figure 6.14, with a magnetic actuator and a capacitance actuator driven by a circuit with applied voltage $V(t)$. We can extend Lagrange's equation to

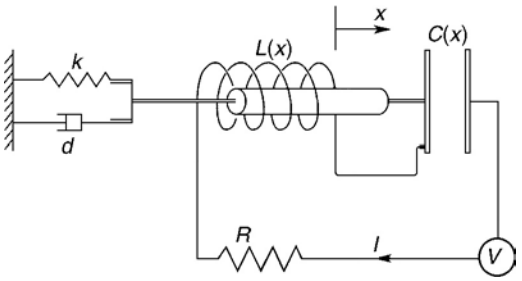



**FIGURE 6.14** Coupled lumped parameter electromechanical system with single degree of freedom mechanical motion $x(t)$.

circuits by defining the charge on the capacitor, $Q$, as another generalized coordinate along with $x$, i.e., in Lagrange's formulation, $q_1 = x$, $q_2 = Q$. Then we add to the kinetic energy function a magnetic energy function $W_m(\dot{Q}, x)$, and add to the potential energy an electric field energy function $W_e(Q, x)$. The equations of both the mass and the circuit can then be derived from

$$\frac{d}{dt}\frac{\partial[T + W_m]}{\partial \dot{q}_k} - \frac{\partial[T + W_m]}{\partial q_k} + \frac{\partial[W + W_e]}{\partial q_k} = Q_k \tag{6.44}$$

The generalized force must also be modified to account for the energy dissipation in the resistor and the energy input of the applied voltage $V(t)$, i.e., $Q_1 = -c\dot{x}$, $Q_2 = -R\dot{Q} + V(t)$. In this example the magnetic energy is proportional to the inductance $L(x)$, and the electric energy function is inversely proportional to the capacitance $C(x)$. Applying Lagrange's equations automatically results in expressions for the magnetic and electric forces as derivatives of the magnetic and electric energy functions, respectively, i.e.,

$$W_m = \frac{1}{2}L(x)\dot{Q}^2 = \frac{1}{2}LI^2, \qquad W_e = \frac{1}{2C(x)}Q^2 \tag{6.45}$$

$$F_m = \frac{\partial W_m(x, \dot{Q})}{\partial x} = \frac{1}{2}I^2\frac{dL(x)}{dx}, \qquad F_e = -\frac{\partial W_e(x, Q)}{\partial x} = -\frac{1}{2}Q^2\frac{d}{dx}\left[\frac{1}{C(x)}\right] \tag{6.46}$$

These remarkable formulii are very useful in that one can calculate the electromagnetic forces by just knowing the dependence of the inductance and capacitance on the displacement $x$. These functions can often be found from electrical measurements of $L$ and $C$.

### Example: Electric Force on a Comb-Drive MEMS Actuator

Consider the motion of an elastically constrained plate between two grounded fixed plates as in a MEMS comb-drive actuator in Figure 6.15. When the moveable plate has a voltage $V$ applied, there is stored electric field energy in the two gaps given by

$$W_e^*(V, x) = \frac{1}{2}\varepsilon_0 V^2 A\frac{d_0}{d_0^2 - x^2} \tag{6.47}$$

In this expression the electric energy function is written in terms of the voltage $V$ instead of the charge on the plates $Q$ as in Equations (6.45) and (6.46). Also the initial gap is $d_0$, and the area of the plate

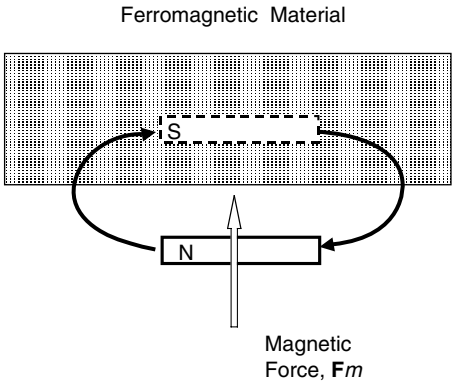



**FIGURE 6.15** Example of electric force on the elements of a comb-drive actuator.

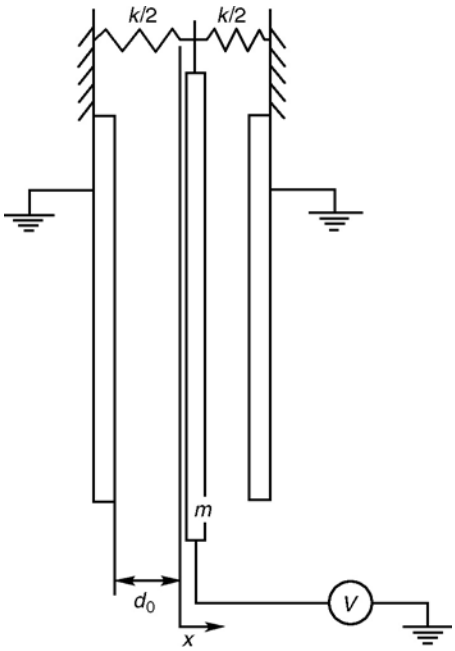



**FIGURE 6.16** Decrease in natural frequency of a MEMS device with applied voltage as an example of negative electric stiffness [From Adams (1996)].

is $A$. Using the force expressions derived from Lagrange's equations (6.44), the electric charge force on the plate is given by

$$F_e = \frac{\partial}{\partial x} W_e^*(V, x) = \frac{\varepsilon_0 V^2 A}{d_0} \frac{x}{(1 - x^2/d^2)^2} \tag{6.48}$$

This expression shows that the electric stiffness is negative for small $x$, which means that the voltage will decrease the natural frequency of the plate. This idea has been applied to a MEMS comb-drive actuator by Adams (1996) in which the voltage could be used to tune the natural frequency of a MEMS accelerometer, as shown in Figure 6.16.

## 6.9 Earnshaw's Theorem and Electromechanical Stability

It is not well known that electric and magnetic forces in mechanical systems can produce static instability, otherwise known as *elastic buckling* or *divergence*. This is a consequence of the inverse square nature of many electric and magnetic forces. It is well known that the electric and magnetic field potential $\Phi$ satisfies Laplace's equation, $\nabla^2\Phi = 0$. There is a basic theorem in potential theory about the impossibility of a relative maximum or minimum value of a potential $\Phi(\mathbf{r})$ for solutions of Laplace's equation except at a boundary. It was stated in a theorem by Earnshaw (1829) that it is impossible for a static set of charges, magnetic and electric dipoles, and steady currents to be in a stable state of equilibrium without mechanical or other feedback or dynamic forces (see, for example, Moon, 1984, 1994).

One example of Earnshaw's theorem is the instability of a magnetic dipole (e.g., a permanent magnet) near a ferromagnetic surface (Figure 6.17). Levitated bearings based on ferromagnetic forces, for example, require feedback control. Earnshaw's theorem also implies that if there is one degree of freedom with stable restoring forces, there must be another degree of freedom that is unstable. Thus the equilibrium positions for a pure electric or magnetic system of charges and dipoles must be saddle points. The implication for the force potentials is that the matrix of second derivatives is not positive definite. For example, suppose there are three generalized position coordinates $\{s_u\}$ for a set of electric charges. Then if the generalized forces are proportional to the gradient of the potential, $\nabla\Phi$, then the generalized electric stiffness matrix $\mathbf{K}_{ij}$, given by

$$\mathbf{K}_{ij} = \left[\frac{\partial^2\Phi}{\partial s_i \partial s_j}\right]$$

will not be positive definite. This means that at least one of the eigenvalues will have negative stiffness.

Another example of electric buckling is a beam in an electric field with charge induced by an electric field on two nearby stationary plates as in Figure 6.15. The induced charge on the beam will be attracted to either of the two plates, but is resisted by the elastic stiffness of the beam. As the voltage is increased, the combined electric and elastic stiffnesses will decrease until the beam buckles to one or the other of the two sides. Before buckling, however, the natural frequency of the charged beam will decrease (Figure 6.16). This property has been observed experimentally in a MEMS device. A similar magneto elastic buckling is observed for a thin ferromagnetic elastic beam in a static magnetic field (see Moon, 1984). Both electroelastic and magnetoelastic buckling are derived from the same principle of Earnshaw's theorem.

There are dramatic exceptions to Earnshaw's stability theorem. One of course is the levitation of 50-ton vehicles with magnetic fields, known as MagLev, or the suspension of gas pipeline rotors using feedback controlled magnetic bearings (see Moon, 1994). Here either the device uses feedback forces, i.e., the fields

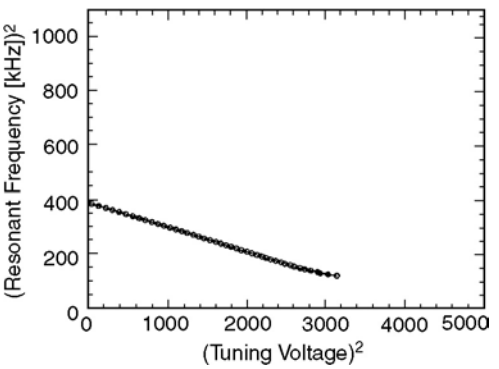



**FIGURE 6.17** Magnetic force on a magnetic dipole magnet near a ferromagnetic half space with image dipole shown.

are not static, or the source of one of the magnetic fields is a superconductor. Diamagnetic forces are exceptions to Earnshaw's theorem, and superconducting materials have properties that behave like diamagnetic materials. Also new high-temperature superconductivity materials, such as YBaCuO, exhibit magnetic flux pinning forces that can be utilized for stable levitation in magnetic bearings without feedback (see Moon, 1994).

## References

Adams, S. G. (1996), *Design of Electrostatic Actuators to Tune the Effective Stiffness of Micro-Mechanical Systems,* Ph.D. Dissertation, Cornell Unversity, Ithaca, New York.

Goldstein, H. (1980), *Classical Mechanics,* Addison-Wesley, Reading, MA.

Jackson, J. D. (1968), *Classical Electrodynamics,* J. Wiley & Sons, New York.

Lee, C. K. and Moon, F. C. (1989), "Laminated piezopolymer plates for bending sensors and actuators," *J. Acoust. Soc. Am.,* **85**(6), June 1989.

Melcher, J. R. (1981), *Continuum Electrodynamics,* MIT Press, Cambridge, MA.

Miu, D. K. (1993), *Mechatronics,* Springer-Verlag, New York.

Moon, F. C. (1984), *Magneto-Solid Mechanics,* J. Wiley & Sons, New York.

Moon, F. C. (1994), *Superconducting Levitation,* J. Wiley & Sons, New York.

Moon, F. C. (1999), *Applied Dynamics,* J. Wiley & Sons, New York.

Yu, Y.-Y. (1996), *Vibrations of Elastic Plates,* Springer-Verlag, New York.

# 7

# Modeling and Simulation for MEMS




Carla Purdy
*University of Cincinnati*


## 7.1 Introduction

Accurate modeling and efficient simulation, in support of greatly reduced development cycle time and cost, are well established techniques in the miniaturized world of integrated circuits (ICs). Simulation accuracies of 5% or less for parameters of interest are achieved fairly regularly [1], although even much less accurate simulations (25–30%, e.g.) can still be used to obtain valuable information [2]. In the IC world, simulation can be used to predict the performance of a design, to analyze an already existing component, or to support automated synthesis of a design. Eventually, MEMS simulation environments should also be capable of these three modes of operation. The MEMS developer is, of course, most interested in quick access to particular techniques and tools to support the system currently under development. In the long run, however, consistently achieving acceptably accurate MEMS simulations will depend both on the ability of the CAD (computer-aided design) community to develop robust, efficient, user-friendly tools which will be widely available both to cutting-edge researchers and to production engineers and on the existence of readily accessible standardized processes. In this chapter we focus on fundamental approaches which will eventually lead to successful MEMS simulations becoming routine.

We also survey available tools which a MEMS developer can use to achieve good simulation results. Many of these tools build MEMS development systems on platforms already in existence for other technologies, thus leveraging the extensive resources which have gone into previous development and avoiding "reinventing the wheel."

For our discussion of modeling and simulation, the salient characteristics of MEMS are:

1. inclusion and interaction of multiple domains and technologies,
2. both two- and three-dimensional behaviors,
3. mixed digital (discrete) and analog (continuous) input, output, and signals, and
4. micro- (or nano-) scale feature sizes.

Techniques for the manufacture of reliable (two-dimensional) systems with micro- or nano-scale feature sizes (Characteristic 4) are very mature in the field of microelectronics, and it is logical to attempt to extend these techniques to MEMS, while incorporating necessary changes to deal with Characteristics 1–3. Here we survey some of the major principles which have made microelectronics such a rapidly evolving field, and we look at microelectronics tools which can be used or adapted to allow us to apply these principles to MEMS. We also discuss why applying such strategies to MEMS may not always be possible.

## 7.2 The Digital Circuit Development Process: Modeling and Simulating Systems with Micro- (or Nano-) Scale Feature Sizes

A typical VLSI digital circuit or system process flow is shown in Figure 7.1, where the dotted lines show the most optimistic point to which the developer must return if errors are discovered. Option A, for a "mature" technology, is supported by efficient and accurate simulators, so that even the first actual implementation ("first silicon") may have acceptable performance. As a process matures, the goal is to have better and better simulations, with a correspondingly smaller chance of discovering major performance flaws after implementation. However, development of models and simulators to support this goal is in itself a major task. Option B (immature technology), at its extreme, would represent an experimental technology for which not enough data are available to support even moderately robust simulations. In modern software and hardware development systems, the emphasis is on tools which provide increasingly good support for the initial stages of this process. This increases the probability that conceptual or design errors will be identified and modifications made as early in the process as possible and thus decreases both development time and overall development cost.

At the microlevel, the development cycle represented by Option A is routinely achieved today for many digital circuits. In fact, the entire process can in some cases be highly automated, so that we have "silicon compilers" or "computers designing computers." Thus, not only design analysis, but even design synthesis is possible. This would be the case for well-established silicon-based CMOS technologies, for example. There are many characteristics of digital systems which make this possible. These include:

- Existence of a small set of basic digital circuit elements. All Boolean functions can be realized by combinations of the logic functions AND, OR, NOT. In fact, all Boolean functions can be realized by combinations of just one gate, a NAND (NOT-AND) gate. So if a "model library" of basic gates (and a few other useful parts, such as I/O pins, multiplexors, and flip-flops) is developed, systems can be implemented just by combining suitable library elements.
- A small set of standardized and well-understood technologies, with well-characterized fabrication processes that are widely available. For example, in the United States, the MOSIS service [3] provides access to a range of such technologies. Similar services elsewhere include CMP in France [4], Europractice in Europe [5], VDEC in Japan [6], and CMC in Canada [7].
- A well-developed educational infrastructure and prototyping facilities. These are provided by all of the services listed above. These types of organization and educational support had their origins in

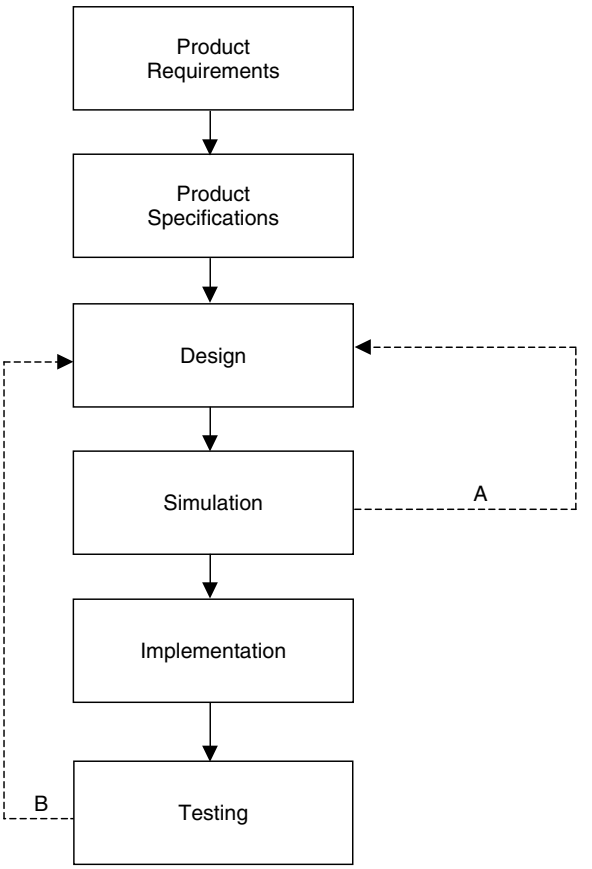



**FIGURE 7.1** Product design process. A: mature technology, B: immature technology.

the work of Mead and Conway [8] and continue to produce increasingly sophisticated VLSI engineers. An important aspect of this infrastructure is that it also provides, at relatively low cost, access to example devices and systems, made with stable fabrication processes, whose behavior can be tested and compared to simulation results, thereby enabling improvements in simulation techniques.

- "Levels and views" (abstraction and encapsulation or "information hiding") (see [9]). This concept is illustrated in Figure 7.2(a). For the VLSI domain, we can identify at least five useful levels of abstraction, from the lowest (layout geometry) to the highest (system specification). We can also "view" a system behaviorally, structurally, or physically. In the *behavioral domain* we describe the functionality of the circuit without specifying how this functionality will be achieved. This allows us to think clearly about what the system needs to do, what inputs are needed, and what outputs will be provided. Thus we can view the component as a "black box" that has specified responses to given inputs. The current through a MOS field effect transistor (MOSFET), given as a function of the gate voltage, is a (low-level) behavioral description, for example. In the *physical domain* we specify the actual physical parts of the circuit. At the lowest levels in this domain, we must choose what material each piece of the circuit will be made from (for example, which pieces of wire will lie in each of the metal layers usually provided in a CMOS circuit) and exactly where each piece will be placed in the actual physical layout. The physical description will be translated directly into mask layouts for the circuit. The *structural domain* is intermediate between physical and behavioral. It provides an interface between the functionality specified in the behavioral domain, which ignores geometry, and the geometry specified in the physical domain, which ignores functionality. In this

(a)

| Levels | Views | | |
|---|---|---|---|
| | Behavioral | Structural | Physical |
| 4 | Performance Specifications | CPUs, Memory, Switches, Controllers, Buses | Physical Partitions |
| 3 | Algorithms | Modules, Data Structures | Clusters |
| 2 | Register Transfers | ALUs, MUXs, Registers | Floorplans |
| 1 | Boolean Equations, FSMs | Gates, Flip-flops | Cells, Modules |
| 0 | Transfer Functions, Timing | Transistors, Wires, Contacts, Vias | Layout Geometry |

(b)

| Levels | Views | | |
|---|---|---|---|
| | Behavioral | Structural | Physical |
| 4 | Performance Specifications | Sensors, Actuators, Systems | Physical Partitions |
| 3 | | Multiple Energy Domain Components | Clusters |
| 2 | | Domain-Domain Components | Floorplans |
| 1 | | Single Energy Domain Components | Cells, Modules |
| 0 | Transfer Functions, Timing | Beams, Membranes, Holes, Grooves, Joints | Layout Geometry |

**FIGURE 7.2** A taxonomy for component development ("levels and views"): (a) standard VLSI classifications, (b) a partial classification for MEMS components.

intermediate domain, we can carry out logic optimization and state minimization, for example. A schematic diagram is an example of a structural description. Of course, not all circuit characteristics can be completely encapsulated in a single one of these views. For example, if we change the physical size of a wire, we will probably affect the timing, which is a behavioral property. The principle of encapsulation leads naturally to the development of extensive IP (intellectual property), i.e., libraries of increasingly sophisticated components that can be used as "black boxes" by the system developer.

- Well-developed models for basic elements that clearly delineate effects due to changes in design, fabrication process, or environment. For example, in [10], the factors in the basic first-order equations for $I_{ds}$, the drain-to-source current in an NMOS transistor, can clearly be divided into those under the control of the designer (W/L, the width-to-length ratio for the transistor channel), those dependent on the fabrication process ($\varepsilon$, the permittivity of the gate insulator, and $t_{ox}$, the thickness of the gate insulator), those dependent on environmental factors ($V_{ds}$ and $V_{gs}$, the drain-to-source and gate-to-source voltages, respectively), and those that are a function of both the fabrication process and the environment ($\mu$, the effective surface mobility of the carriers in the channel, and $V_t$, the threshold voltage). More detailed information on modeling MOSFETs can be found in [11]. Identification of fundamental parameters in one stage of the development process can be of great value in other stages. For example, the minimum feature size $\lambda$ for a given technology can be used to develop a set of "design rules" that express mandatory overlaps and spacings for the different physical materials. A design tool can then be developed to "enforce" these rules, and the consequences can be used to simplify, to some extent, the modeling and simulation stages. The parameter $\lambda$ can also be used to express effects due to scaling when scaling is valid.

- Mature tools for design and simulation, which have evolved over many generations and for which moderately priced versions are available from multiple sources. For example, many of today's tools incorporate versions of the design tool MAGIC [12] and the simulator SPICE (Simulation Program with Integrated Circuit Emphasis) [13], both of which were originally developed at the University of California, Berkeley. Versions of the SPICE simulator typically support several device models (currently, for example, six or more different MOS models and five different transmission line models), so that a developer can choose the level of device detail appropriate to the task at hand. Free or low-cost versions of both MAGIC and SPICE, as well as extended versions of both tools, are widely available. Many different techniques, such as model binning (optimizing models for specific ranges of model parameters) and inclusion of proprietary process information, are employed to produce better models and simulation results, especially in the HSPICE version of SPICE and in other high-end versions of these tools [11].
- Integrated development systems that are widely available and that provide support for a variety of levels and views, extensive component libraries, user-friendly interfaces and online help, as well as automatic translation between domains, along with error and constraint checking. In an integrated VLSI development system, sophisticated models, simulators, and translators keep track of circuit information for multiple levels and views, while allowing the developer to focus on one level or view at a time. Many development systems available today also support, at the higher levels of abstraction, structured "programming" languages such as VHDL (Very Large Scale Integrated Circuit Hardware Description Language) [14,15] or Verilog [16].

A digital circuit developer has many options, depending on performance constraints, number of units to be produced, desired cost, available development time, etc. At one extreme the designer may choose to develop a "custom" circuit, creating layout geometries, sizing individual transistors, modeling RC effects in individual wires, and validating design choices through extensive low-level SPICE-based simulations. At the other extreme, the developer can choose to produce a PLD (programmable logic device), with a predetermined basic layout geometry consisting of cells incorporating programmable logic and storage (Figure 7.3) that can be connected as needed to produce the desired device functionality. A high end PLD may contain as many as 100,000 (100 K) cells similar to the one in Figure 7.3 and an additional 100 K bytes of RAM (random access memory) storage. In an integrated development system, such as

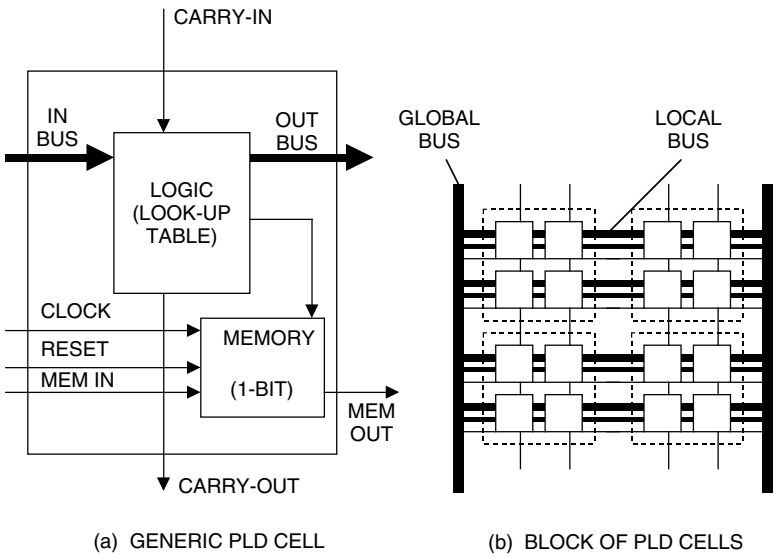



**FIGURE 7.3** A generic programmable logic device architecture.

those provided by [17] and [18], the developer enters the design in either schematic form or a high level language, and then the design is automatically "compiled" and mapped to the PLD geometry, and functional and timing simulations can be run. If the simulation results are acceptable, an actual PLD can then be programmed directly, as a further step in the development process, and even tested, to some extent, with the same set of test data as was used for the simulation step. This "rapid prototyping" [19] for the production of a "chip" is not very different from the production of a working software program (and the PLD can be reprogrammed if different functionality is later desired). Such a system, of course, places many constraints on achievable designs. In addition, the automated steps, which rely on heuristics rather than exact techniques to find acceptable solutions to the many computationally complex problems that need to be solved during the development process, sacrifice performance for ease of development, so that a device designed in such a system will never achieve the ultimate performance possible for the given technology. However, the trade-offs include ease of use, much shorter development times, and the management of much larger numbers of individual circuit elements than would be possible if each individual element were tuned to its optimum performance. In addition, if a high-level language is used for input, an acceptable design can often be translated, with few changes, to a more powerful design system that will allow implementation in more flexible technologies and additional fine tuning of circuit performance. In Figure 7.4 we see some of the levels of abstraction which are present in such a

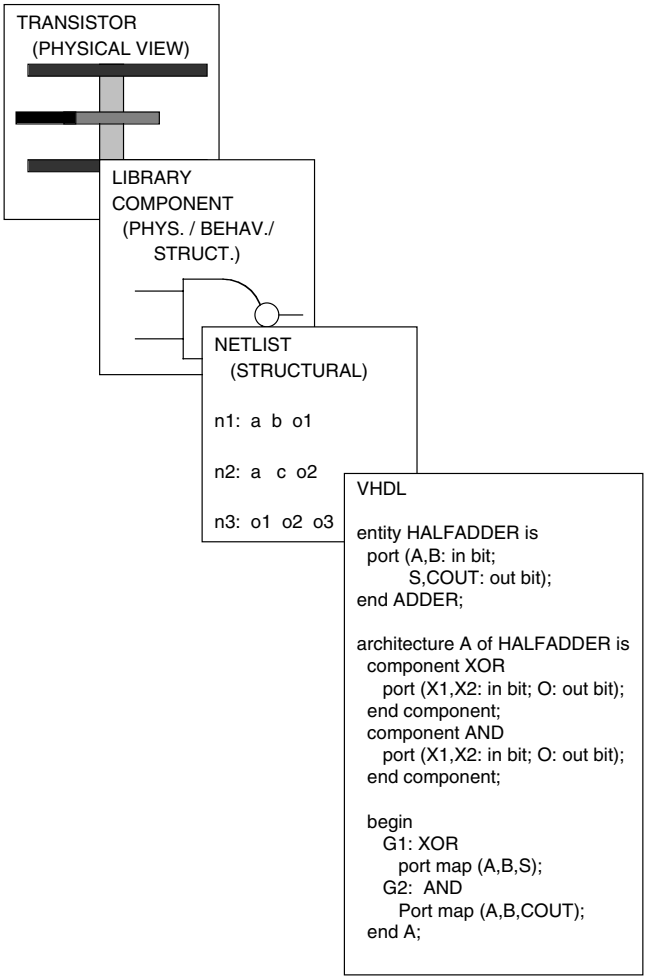



**FIGURE 7.4** Levels of abstraction–half adder.

development process, with the lowest level being detailed transistor models and the highest a VHDL description of a half adder.

## 7.3 Analog and Mixed-Signal Circuit Development: Modeling and Simulating Systems with Micro- (or Nano-) Scale Feature Sizes and Mixed Digital (Discrete) and Analog (Continuous) Input, Output, and Signals

At the lowest level, digital circuits are in fact analog devices. A CMOS inverter, for example, does not "switch" instantaneously from a voltage level representing binary 0 to a voltage level representing binary 1. However, by careful design of the inverter's physical structures, it is possible to make the switching time from the range of voltage outputs which are considered to be "0" to the range considered to be "1" (or vice versa) acceptably short. In MOSFETs, for example, the two discrete signals of interest can be identified with the transistor, modeled as a switch, being "open" or "closed," and the "switching" from one state to another can be ignored except at the very lowest levels of abstraction. In much design and simulation work, the analog aspects of the digital circuit's behavior can thus be ignored. Only at the lower levels of abstraction will the analog properties of VLSI devices or the quantum effects occurring, e.g., in a MOSFET need to be explicitly taken into account, ideally by powerful automated development tools supported by detailed models. At higher levels this behavior can be encapsulated and expressed in terms of minimum and maximum switching times with respect to a given capacitive load and given voltage levels. Even in digital systems, however, as submicron feature sizes become more common, more attention must be paid to analog effects. For example, at small feature sizes, wire delay due to RC effects and crosstalk in nearby wires become more significant factors in obtaining good simulation results [20]. It is instructive to examine how simulation support for digital systems can be extended to account for these factors.

Typically, analog circuit devices are much more likely to be "hand-crafted" than digital devices. SPICE and SPICE-like simulations are commonly used to measure performance at the level of transistors, resistors, capacitors, and inductors. For example, due to the growing importance of wireless and mobile computing, a great deal of work in analog design is currently addressing the question of how to produce circuits (digital, analog, and mixed-signal) that are "low-power," and simulations for devices to be used in these circuits are typically carried out at the SPICE level. Unless a new physical technology is to be employed, the simulations will mostly rely on the commonly available models for transistors, transmission lines, etc., thus encapsulating the lowest level behaviors.

Let us examine the factors given above for the success of digital system simulation and development to see how the analog domain compares. We assume a development cycle similar to that shown in Figure 7.1.

- Is there a small set of basic circuit elements? In the analog domain it is possible to identify sets of components, such as current mirrors, op-amps, etc. However, there is no "universal" gate or small set of gates from which all other devices can be made, as is true in the digital domain. Another complicating factor is that elementary analog circuit elements are usually defined in terms of physical performance. There is no clean notion of 0/1 behavior. Because analog signals are continuous, it is often much more difficult to untangle complex circuit behaviors and to carry out meaningful simulations where clean parameter separations give clear results. Once a preliminary analog device or circuit design has been developed, the process of using simulations to decide on exact parameter values is known as "exploring the design space." This process necessarily exhibits high computational complexity. Often heuristic methods such as simulated annealing, neural nets, or a genetic algorithm can be used to perform the necessary search efficiently [21].
- Is there a small set of well-understood technologies? In this area, the analog and mixed signal domain is similar to the digital domain. Much analog development activity focuses on a few standard and well-parameterized technologies. In general, analog devices are much more

sensitive to variations in process parameters, and this must be accounted for in analog simulation. Statistical techniques to model process variation have been included, for example, in the APLAC tool [22], which supports object-oriented design and simulation for analog circuits. Modeling and simulation methods, which incorporate probabilistic models, will become increasingly important as nanoscale devices become more common and as new technologies depending on quantum effects and biology-based computing are developed. Several current efforts, for example, are aimed at developing a "BIOSPICE" simulator, which would incorporate more stochastic system behavior [23].

- Is there a well-developed educational infrastructure and prototyping facilities? All the organizations, which support education and prototyping in the digital domain [3–7], provide similar support for analog and mixed-signal design.
- Are encapsulation and abstraction widely employed? In the past few years, a great deal of progress has been made in incorporating these concepts into analog and mixed-signal design systems. The wide availability of very powerful computers, which can perform the necessary design and simulation tasks in reasonable amounts of time, has helped to make this progress possible. In [24], for example, top-down, constraint-driven methods are described, and in [25] a rapid prototyping method for synthesizing analog and mixed signal systems, based on the tool suite VASE (VHDL-AMS Synthesis Environment), is demonstrated. These methods rely on classifications similar to those given for digital systems in Figure 7.2(a).
- Are there well-developed models, mature tools, and integrated development systems which are widely available? In the analog domain, there is still much more to be done in these areas than in the digital domain, but prototypes do exist. In particular, the VHDL and Verilog languages have been extended to allow for analog and mixed-signal components. The VHDL extension, e.g., VHDL-AMS [14], will allow the inclusion of any algebraic or ordinary differential equation in a simulation. However, there does not exist a completely functional VHDL-AMS simulator, although a public domain version, incorporating many useful features, is available at [26] and many commercial versions are under development (e.g., [27]). Thus, at present, expanded versions of MAGIC and SPICE are still the most widely-used design and simulation tools. While there have been some attempts to develop design systems with configurable devices similar to the digital devices shown in Figure 7.3, these have not so far been very successful. Currently, more attention is being focused on component-based development with design reuse for SOC (systems on a chip) through initiatives such as [28].

## 7.4 Basic Techniques and Available Tools for MEMS Modeling and Simulation

Before trying to answer the above questions for MEMS, we need to look specifically at the tools and techniques the MEMS designer has available for the modeling and simulation tasks. As pointed out in [29,30], the bottom line is, in any simulator, all models are not created equal. The developer must be very clear about what parameters are of greatest interest and then must choose the models and simulation techniques (including implementation in a tool or tools) that are most likely to give the most accurate values for those parameters in the least amount of simulation time. For example, the model used to determine static behavior may be different from the model needed for an adequate determination of dynamic behavior. Thus, it is useful to have a range of models and techniques available.

### Basic Modeling and Simulation Techniques

We need to make the following choices:

- What kind of behavior are we interested in? IC simulators, for example, typically support DC operating analysis, DC sweep analysis (stepping current or voltage source values) and transient sweep analysis (stepping time values), along with several other types of transient analysis [30].

- Will the computation be symbolic or numeric?
- Will use of an exact equation, nodal analysis, or finite element analysis be most appropriate? Currently, these are the techniques which are favored by most MEMS developers.

To show what these choices entail, let us look at a simple example that combines electrical and mechanical parts. The cantilever beam in Figure 7.5(a), fabricated in metal, polysilicon, or a combination, may be combined with an electrically isolated plate to form a parallel plate capacitor. If a mechanical force or a varying voltage is applied to the beam (Figure 7.5(b1)), an accelerometer or a switch can be obtained [31]. If instead the plate can be moved back and forth, a more efficient accelerometer design results (Figure 7.5(b2)); this is the basic design of Analog Devices' accelerometer, probably the first truly successful commercial MEMS device [32,33]. If several beams are combined into two "combs," a comb-drive sensor or actuator results, as in Figure 7.5(b3) [34]. Let us consider just the simplest case, as shown in Figure 7.5(b1).

If we assume the force on the beam is concentrated at its end point, then we can use the method of [35] to calculate the "pull-in" voltage, i.e., the voltage at which the plates are brought together, or to a stopper which keeps the two plates from touching. We model the beam as a dampened spring-mass system and look for the force $F$, which, when translated into voltage, will give the correct $x$ value for the beam to be "pulled in."

$$F = mx'' + Bx' + kx$$

Here mass $m = \rho WTL$, where $\rho$ is the density of the beam material, $I = WT^3/12$ is the moment of inertia, $k = 3EI/L^3$, $E$ is the Young's modulus of the beam material, and $B = (k/EI)^{1/4}$. This second-order linear differential equation can be solved numerically to obtain the pull-down voltage. In this case, since a closed form expression can be obtained for $x$, symbolic computation would also be an option. In [36] it is shown that for this simple problem several commonly used methods and tools will give the same result, as is to be expected.

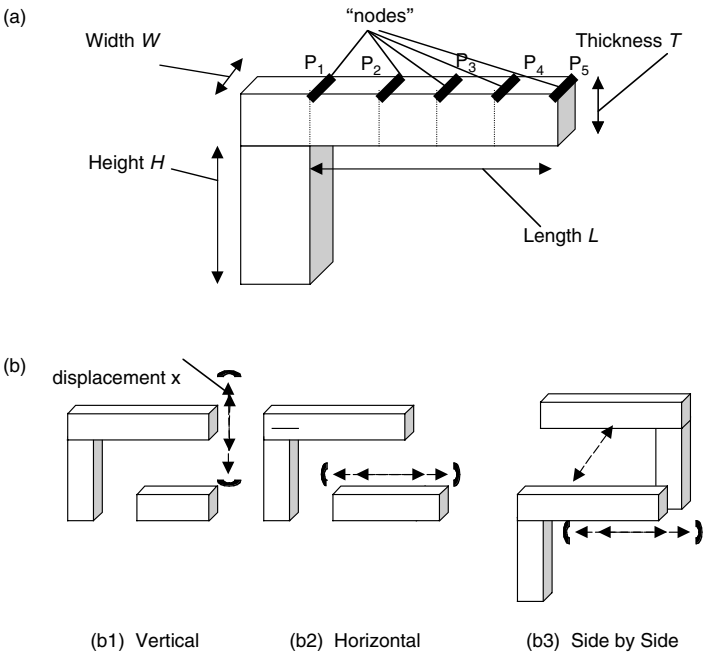
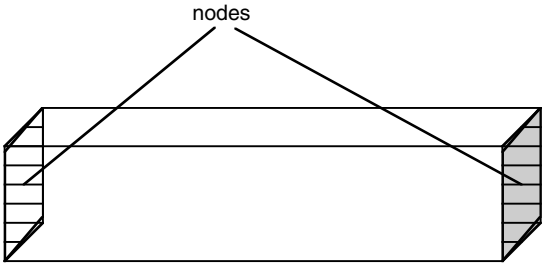



**FIGURE 7.5** Cantilever beam and beam–capacitor options: (a) cantilever beam dimensions, (b) basic beam–capacitor designs.

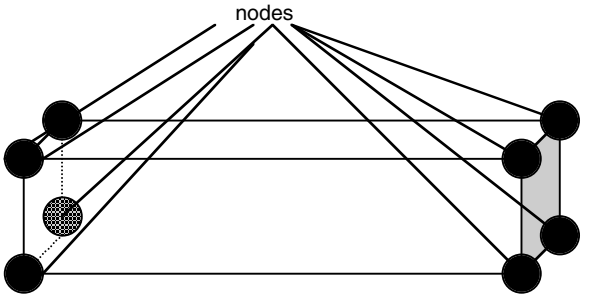
To obtain a more accurate model of the beam we can use the method of nodal analysis, that treats the beam as a graph consisting of a set of edges or "devices," linked together at "nodes." Nodal analysis assumes that at equilibrium the sum of all values around each closed loop (the "across" quantities) will be zero, as will the sum of all values entering or leaving a given node (the "through" quantities). Thus, for example, the sum of all forces and moments on each node must be zero, as must the sum of all currents flowing into or out of a given node. This type of modeling is sometimes referred to as "lumped parameter," since quantities such as resistance and capacitance, which are in fact distributed along a graph edge, are modeled as discrete components. In the electrical domain Kirchhoff's laws are examples of these rules. This method, which is routinely applied to electrical circuits in elementary network analysis courses (see, e.g., [37]), can easily be applied to other energy domains by using correct domain equivalents (see, e.g., [38]). A comprehensive discussion of the theory of nodal analysis can be found in [39]. In Figure 7.5(a), the cantilever beam has been divided into four "devices," subbeams between node $i$ and $i + 1$, $i = 1, 2, 3, 4$, where the positions of nodes $i$ and $i + 1$ are described by $(x_i, y_i, \theta_i)$ and $(x_{i+1}, y_{i+1}, \theta_{i+1})$ the coordinates and slope at $P_i$ and $P_{i+1}$. The beam is assumed to have uniform width $W$ and thickness $T$, and each subbeam is treated as a two-dimensional structure free to move in three-space. In [40] a modified version of nodal analysis is used to develop numerical routines to simulate several MEMS behaviors, including static and transient behavior of a beam-capacitor actuator. This modified method also adds position coordinates $z_i$ and $z_{i+1}$ and replaces the slope $\theta_i$ at each node with a vector of slopes, $\theta_{ix}$, $\theta_{iy}$, and $\theta_{iz}$, giving each node six degrees of freedom.

Since nodal analysis is based on linear elements represented as the edges in the underlying graph, it cannot be used to model many complex structures and phenomena such as fluid flow or piezoelectricity. Even for the cantilever beam, if the beam is composed of layers of two different materials (e.g., polysilicon and metal), it cannot be adequately modeled using nodal analysis. The technique of finite element analysis (FEA) must be used instead. For example, in some follow-up work to that reported in [36], nodal analysis and symbolic computation gave essentially the same results, but the FEA results were significantly different. Finite element analysis for the beam begins with the identification of subelements, as in Figure 7.5(a), but each element is treated as a true three-dimensional object. Elements need not all have the same shape, for example, tetrahedral and cubic "brick" elements could be mixed together, as appropriate. In FEA, one cubic element now has eight nodes, rather than two (Figure 7.6), so computational complexity is increased. Thus, developing efficient computer software to carry out FEA for a given structure can be a difficult task in itself. But this general method can take into account many features that cannot be adequately addressed using nodal analysis, including, for example, unaligned beam sections, and surface texture (Figure 7.7). FEA, which can incorporate static, transient, and dynamic behavior, and which can treat heat and fluid flow, as well as electrical, mechanical, and other forces, is explained in detail in [41]. The basic procedure is as follows:

- Discretize the structure or region of interest into finite elements. These need not be homogeneous, either in size or in shape. Each element, however, should be chosen so that no sharp changes in geometry or behavior occur at an interior point.
- For each element, determine the element characteristics using a "local" coordinate system. This will represent the equilibrium state (or an approximation if that state cannot be computed exactly) for the element.
- Transform the local coordinates to a global coordinate system and "assemble" the element equations into one (matrix) equation.
- Impose any constraints implied by restricted degrees of freedom (e.g., a fixed node in a mechanical problem).
- Solve (usually numerically) for the nodal unknowns.
- From the global solution, calculate the element resultants.

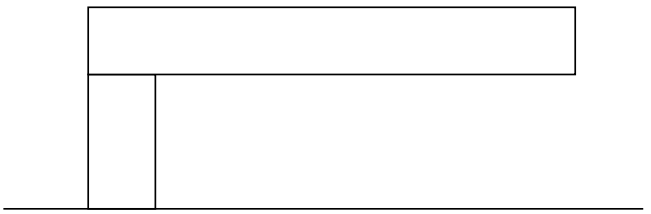



(a) Nodal analysis/Modified nodal analysis
("Linear" elements)

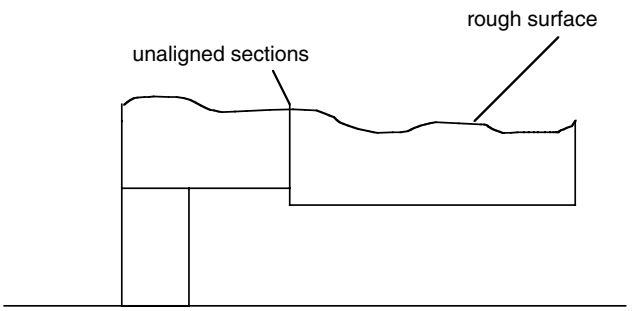



(b) Finite element analysis
(Three-dimensional elements)

**FIGURE 7.6** Nodal analysis and finite element analysis.



(a) Ideal beam




(b) Actual beam

**FIGURE 7.7** Ideal and actual cantilever beams (side view).

## A Catalog of Resources for MEMS Modeling and Simulation

To make our discussion of the state-of-the-art of MEMS simulation less confusing, we first list some of the tools and products available. This list is by no means comprehensive, but it will provide us with a range of approaches for comparison. It should be noted that this list is accurate as of July 2001, but the MEMS development community is itself developing, with both commercial companies and university research sites frequently taking on new identities and partners and also expanding the range of services they offer.

### A. Widely Available Tools for General Numeric and Symbolic Computation

These tools are relatively easy to learn to use. Most engineering students will have mastered at least one before obtaining a bachelor's degree. They can be used to model a device "from scratch" and to perform simple simulations. For more complex simulations, they are probably not appropriate for two reasons. First, neither is optimized to execute long computations efficiently. Second, developing the routines necessary to carry out a complex nodal or finite element analysis will in itself be a time-consuming task and will in most cases only replicate functionality already available in other tools listed here.

- Mathematica [42]. In [36] Mathematica simulation results for a cantilever beam-capacitor system are compared with results from several other tools.
- Matlab (integrated with Maple) [43]. In [44], for example, Matlab simulations are shown to give good approximations for a variety of parameters for microfluidic system components.

### B. Tools Originally Developed for Specific Energy Domains

Low-cost easy to use versions of some of these tools (e.g., SPICE, ANSYS) are also readily available. Phenomena from other energy domains can be modeled using domain translation.

- SPICE (analog circuits) [13]. SPICE is the de facto standard for analog circuit simulators. It is also used to support simulation of transistors and other components for digital systems. SPICE implements numerical methods for nodal analysis. Several authors have used SPICE to simulate MEMS behavior in other energy domains. In [35], for example, the equation for the motion of a damped spring, which is being used to calculate pull-in voltage, is translated into the electrical domain and reasonable simulation accuracy is obtained. In [45] steady-state thermal behavior for flow-rate sensors is simulated by dividing the device to be modeled into three-dimensional "bricks," modeling each brick as a set of thermal resistors, and translating the resulting conduction and convection equations into electrical equivalents.
- APLAC [22]. This object-oriented analog and mixed-signal simulator incorporates routines, which allow statistical modeling of process variation.
- VHDL-AMS [14,26,27]. The VHDL-AMS language, designed to support digital, analog, and mixed-signal simulation, will in fact support simulation of general algebraic and ordinary differential equations. Thus mixed-energy domain simulations can be carried out. VHDL-AMS, which is typically built on a SPICE kernel, uses the technique of nodal analysis. Some VHDL-AMS MEMS models have been developed (see, e.g., [46,47]). Additional information about VHDL-AMS is available at [48].
- ANSYS [49]. Student versions of the basic ANSYS software are widely available. ANSYS is now partnering with MemsPro (see below). ANSYS models both mechanical and fluidic phenomena using FEA techniques. A survey of the ANSYS MEMS initiative can be found at [50].
- CFD software [51]. This package, which also uses FEA, was developed to model fluid flow and temperature phenomena.

### C. Tools Developed Specifically for MEMS

The tools in this category use various simplifying techniques to provide reasonably accurate MEMS simulations without all the computational overhead of FEA.

- SUGAR [40,52]. This free package is built on a Matlab core. It uses nodal analysis and modified nodal analysis to model electrical and mechanical elements. Mechanical elements must be built from a fixed set of components including beams and gaps.
- NODAS v 1.4 [53]. This downloadable tool provides a library of parameterized components (beams, plate masses, anchors, vertical and horizontal electrostatic comb drives, and horizontal electrostatic gaps) that can be interconnected to form MEMS systems. The tool outputs parameters that can be used to perform electromechanical simulations with the Saber simulator [27]. A detailed example is available at [54], and a description of how the tool works (for v 1.3) is also available [55]. Useful information is also available in [70].

### D. "Metatools" Which Attempt to Integrate Two or More Domain-Specific Tools into One Package

- MEMCAD, currently being supported by the firm Coventor [56]. This product was previously supported by Microcosm, Inc. It provides low-level simulation capability by integrating domain-specific FEA tools into one package to support coupled energy domain simulations. It also supports process simulation. Much of the extensive research underlying this tool is summarized in [57].
- MemsPro [58], which currently incorporates links to ANSYS. MemsPro itself is an offshoot of Tanner Tools, Inc. [59], which originally produced a version of MAGIC [12] that would run on PCs. The MemsPro system provides integrated design and simulation capability. Process "design rules" can be defined by the user. SPICE simulation capability is integrated into the toolset, and a data file for use with ANSYS can also be generated. MemsPro does not do true energy domain coupling at this time. Some library components are also available.

### E. Other Useful Resources

- The MEMS Clearinghouse website [60]. This website contains links to products, research groups, and conference information. One useful link is the Material Properties database [61], which includes results from a wide number of experiments by many different research groups. Information from this database can be used for initial "back of the envelope" calculations for component feasibility, for example.
- The Cronos website [62]. This company provides prototyping and production-level fabrication for all three process approaches (surface micromachining, bulk micromachining, and high aspect ratio manufacturing). It is also attempting to build a library of MEMS components for both surface micromachining (MUMPS, or the Multi-User MEMS Process [63]) and bulk micromachining.

## 7.5 Modeling and Simulating MEMS, i.e., Systems with Micro- (or Nano-) Scale Feature Sizes, Mixed Digital (Discrete) and Analog (Continuous) Input, Output, and Signals, Two- and Three-Dimensional Phenomena, and Inclusion and Interaction of Multiple Domains and Technologies

In preceding sections we briefly described the current state-of-the-art in modeling and simulation in both the digital and analog domains. While the digital tools are much more developed, in both the digital and analog domains there exist standard, well-characterized technologies, standard widely available tools, and stable educational and prototyping programs. In the much more complex realm of MEMS, this is not the case. Let us compare MEMS, point by point, with digital and analog circuits.

- Is there a small set of basic elements? The answer to this question is emphatically no. Various attempts have been made by researchers to develop a comprehensive basic set of building blocks, beginning with Petersen's identification of the fundamental component set consisting of beams, membranes, holes, grooves, and joints [64]. Most of these efforts focus on adding mechanical and electromechanical elements. In the SUGAR system, for example, the basic elements are the *beam* and the *electrostatic gap*. In the Carnegie Mellon tool MEMSYN [65], which is supported by the NODAS simulator, basic elements include beams and gaps, as well as plate masses, anchors, and electrostatic comb drives (vertical and horizontal). For the MUMPS process there is the Consolidated Micromechanical Element Library (CaMEL), which contains both a nonparameterized cell database and a library of parameterized elements (which can be accessed through a component "generator," but not directly by the user). CaMEL supports the creation of a limited set of components, including motors and resonators, in a fixed surface-micromachined technology. But the bottom line for MEMS is that no set of basic building blocks has yet been identified which can support all the designs, in many different energy domains and in a variety of technologies, which researchers are interested in building. Moreover, there is no consensus as to how to effectively limit design options so that such a fundamental set could be identified. In addition, the continuous nature of most MEMS behavior presents the same kinds of difficulties that are faced with analog elements. Development of higher level component libraries, however, is a fairly active field, with, for example, ANSYS, CFD, MEMCAD, Carnegie Mellon, and MemsPro all providing libraries of previously designed and tested components for systems developers to use. Most of these components are in the electromechanical domain. As mentioned above, a few VHDL-AMS models are also available, but these will not be of practical value until more robust and complete VHDL-AMS simulators are developed and more experimental results can be obtained to validate these models.
- Is there a small set of well-understood technologies? Again the answer must be no. Almost all digital and analog circuits are essentially two-dimensional, but, in the case of MEMS, many designs can be developed either in the "2.5-dimensional" technology known as micromachining or in the true three-dimensional technology known as bulk micromachining. Thus, before doing any modeling or simulation, the MEMS developer must first choose not only among very different fabrication techniques but also among actual processes. Both the Carnegie Mellon and Cronos tools, for example, are based on processes that are being developed in parallel with the tools. MOSIS does provide central access to technology in which all but the final steps of surface micromachining can be done, but no other centrally maintained processing is available to the community of MEMS researchers in general. For surface micromachining, the fact that the final processing steps are performed in individual research labs is problematic for producing repeatable experimental results. For bulk micromachining examples, fabrication in small research labs rather than in a production environment is more the norm than the exception, so standardization for bulk processes is difficult to achieve. In addition, because much MEMS work is relatively low-volume, most processes are not well enough characterized for low-level modeling to be very effective. In such circumstances it is very difficult to have reliable process characterizations on which to build robust models.
- Is there a well-developed educational infrastructure and prototyping facilities? Again we must answer no. Introductory MEMS courses, especially, are much more likely to emphasize fabrication techniques than modeling and simulation. In [66] a set of teaching modules for a MEMS course emphasizing integrated design and simulation is described. However, this course requires the use of devices previously fabricated for validating design and simulation results, rather than expecting students to complete the entire design-simulate-test-fabricate sequence in one quarter or semester. In addition, well-established institutional practices make it difficult to provide the necessary support for multidisciplinary education which MEMS requires.
- Are encapsulation and abstraction widely employed? In the 1980s many researchers believed that multiple levels of abstraction were not useful for MEMS devices. Currently, however, the concept

| Simulation Tool | Levels Supported |
|---|---|
| Mathematica, Matlab | all |
| MEMCAD | low |
| SPICE | low to medium |
| APLAC | low to medium |
| ANSYS, CFD | low to medium |
| SUGAR, NODAS | low to medium |
| MemsPro | low to medium |
| VHDL-AMS | medium to high |

*Because MEMCAD incorporates process simulations, it supports both physical and behavioral views. All other tools support the behavioral view.

**FIGURE 7.8** Available MEMS simulation tools, by level and view.

of intermediate-level "macromodels" has gained much support [57,70], and increasing emphasis is being placed on developing macromodels for MEMS components that will be a part of larger systems. In addition, there are several systems in development that are based on sets of more primitive components. But this method of development is not the norm, in large part because of the rich set of possibilities inherent in MEMS in general. In Figure 7.2(b) we have given a partial classification of MEMS corresponding to the classification for digital devices in Figure 7.2(a). At this point it is not clear what the optimum number of levels of abstraction for MEMS would be. In Figure 7.8 we have attempted to classify some of the tools from Section 7.4 in terms of their ability to support various levels (since these are simulators, they all support the "behavioral" view. MEMCAD, which allows fabrication process simulation, also supports the "physical" view). Note that VHDL-AMS is the only tool, besides the general-purpose Mathematica and Matlab, that supports a high-level view of MEMS.

- Are there well-developed models, mature tools, and integrated development systems which are widely available? While such systems do not currently exist, it is predicted that some examples should become available within the next ten years [57].

## 7.6 A "Recipe" for Successful MEMS Simulation

A useful set of guidelines for analog simulation can be found in [67]. From this we can construct a set of guidelines for MEMS simulation.

1. Be sure you have access to the necessary domain-specific knowledge for all energy domains of interest before undertaking the project.
2. Never use a simulator unless you know the range of answers beforehand.
3. Never simulate more of the system than is necessary.
4. Always use the simplest model that will do the job.
5. Use the simulator exactly as you would do the experiment.
6. Use a specified procedure for exploring the design space. In most cases this means that you should change only one parameter at a time.
7. Understand the simulator you are using and all the options it makes available.
8. Use the correct multipliers for all quantities.
9. Use common sense.
10. Compare your results with experiments and make them available to the MEMS community.
11. Be sensitive to the possibility of microlevel phenomena, which may make your results invalid.

The last point is particularly important. Many phenomena, which can be ignored at larger feature sizes, will need to be taken into account at the micro level. For example, at the micro scale, fluid flow can behave in dramatically different ways [44]. Many other effects of scaling feature sizes down to the microlevel, including an analysis of why horizontal cantilever beam actuators are "better" than vertical cantilever beam actuators, are discussed in Chapter 9 of [68]. Chapters 4 and 5 of [68] also provide important information for low-level modeling and simulation.

## 7.7 Conclusion: Continuing Progress in MEMS Modeling and Simulation

In the past fifteen years, much progress has been made in providing MEMS designers with simulators and other tools which will give them the ability to make MEMS as useful and ubiquitous as was predicted in [64]. While there is still much to be done, the future is bright for this flexible and powerful technology. One of the main challenges remaining for modeling and simulation is to complete the design and development of a high-level MEMS description language, along with supporting models and simulators, both to speed prototyping and to provide a common user-friendly language for designers. One candidate for such a language is VHDL-AMS. In [69], the strengths and weaknesses of VHDL-AMS as a tool for MEMS development are discussed. Strengths include the ability to handle both discrete and continuous behavior, smooth transitions between levels of abstraction, the ability to handle both conservative and nonconservative systems simultaneously, and the ability to import code from other languages. Major drawbacks include the inability to do symbolic computation, the limitation to ordinary differential equations, lack of support for frequency domain simulations, and inability to do automatic unit conversions. It remains to be seen whether VHDL-AMS will eventually be extended to make it more suitable to support the MEMS domain. But it is highly likely that VHDL-AMS or some similar language will eventually come to be widely used and appreciated in the MEMS community.

### References


1. Kielkowski, R.M., *SPICE: Practical Device Modeling,* McGraw-Hill, 1995.
2. Leong, S.K., Extracting MOSFET RF SPICE models, http://www.polyfet.com/MTT98.pdf (accessed July 20, 2001).
3. http://www.mosis.edu (accessed July 20, 2001).
4. http://cmp.imag.fr (accessed July 20, 2001).
5. http://www.imec.be/europractice/europractice.html (accessed July 20, 2001).
6. http://www.vdec.u-tokyo.ac.jp/English (accessed July 20, 2001).
7. http://www.cmc.ca (accessed July 20, 2001).
8. Mead, C. and Conway, L., *Introduction to VLSI Systems,* Addison-Wesley, 1980.
9. Gajski, D. and Thomas, D., Introduction to silicon compilation, in *Silicon Compilation,* D. Gajski, Ed., Addison-Wesley, 1988, 1–48.
10. Weste, N. and Esraghian, K., *Principles of CMOS VLSI Design: A Systems Perspective,* 2nd ed., Addison-Wesley, 1993.
11. Foty, D., *MOSFET Modeling with SPICE,* Prentice Hall, 1997.
12. http://www.research.compaq.com/wrl/projects/magic/magic.html (accessed July 20, 2001).
13. http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE (accessed July 20, 2001).
14. Design Automation Standards Committee, IEEE Computer Society, *IEEE VHDL Standard Language Reference Manual (Integrated with VHDL-AMS Changes),* Standard 1076.1, IEEE, 1997.
15. Ashenden, P., *The Designer's Guide to VHDL, 2nd ed.,* Morgan Kauffman, 2001.
16. Bhasker, J., *A Verilog HDL Primer,* 2nd ed., Star Galaxy Pub., 1999.
17. http://www.altera.com (accessed July 20, 2001).
18. http://www.xilinx.com (accessed July 20, 2001).

19. Hamblen, J.O. and Furman, M.D., *Rapid Prototyping of Digital Systems, A Tutorial Approach,* Kluwer, 1999.
20. Uyemura, J.P., *Introduction to VLSI Circuits and Systems,* John Wiley & Sons, Inc., 2002.
21. Sobecks, B., Performance Modeling of Analog Circuits via Neural Networks: The Design Process View, Ph.D. Dissertation, University of Cincinnati, 1998.
22. http://www.aplac.hut.fi (accessed July 20, 2001).
23. Weiss, R., Homsy, G., and Knight, T., Toward *in vivo* digital circuits, http://www.swiss.ai.mit.edu/~rweiss/bio-programming/dimacs99-evocomp-talk/ (accessed July 20, 2001).
24. Chang, H., Charbon, E., Choudhury, U., Demir, A., Liu, Felt E., Malavasi, E., Sangiovanni-Vincentelli, A., Charbon, E., and Vassiliou, I., *A Top-down, Constraint-Driven Design Methodology for Analog Integrated Circuits,* Kluwer Academic Publishers, 1996.
25. Ganesan, S., Synthesis and Rapid Prototyping of Analog and Mixed Signal Systems, Ph.D. Dissertation, University of Cincinnati, 2001.
26. SEAMS simulator project, University of Cincinnati ECECS Department, Distributed Processing Laboratory, http://www.ececs.uc.edu/~hcarter (accessed July 20, 2001).
27. http://www.analogy.com/products/Simulation/simulation.htm#Saber (accessed July 20, 2001).
28. www.design-reuse.com (accessed July 20, 2001).
29. S. M. Sandler and Analytical Engineering Inc., *The SPICE Handbook of 50 Basic Circuits,* http://dacafe.ibsystems.com/DACafe/EDATools/EDAbooks/SpiceHandBook (accessed July 20, 2001).
30. Kielkowski, R.M., *Inside Spice,* 2nd ed., McGraw Hill, 1998.
31. Gibson, D., Hare, A., Beyette, F., Jr., and Purdy, C., Design automation of MEMS systems using behavioral modeling, *Proc. Ninth Great Lakes Symposium on VLSI,* Ann Arbor Mich. (Eds. R.J. Lomax and P. Mazumder), March 1999, pp. 266–269.
32. http://www.analog.com/industry/iMEMS (accessed July 20, 2001).
33. http://www-ccrma.stanford.edu/CCRMA/Courses/252/sensors/node6.html (accessed July 20, 2001).
34. Tang, W., Electrostatic Comb Drive for Resonant Sensor and Actuator Applications, Ph.D. Dissertation, UC Berkeley, 1990.
35. Lo, N.R., Berg, E.C., Quakkelaar, S.R., Simon, J.N., Tachiki, M., Lee, H.-J., and Pister, S.J., Parameterized layout synthesis, extraction, and SPICE simulation for MEMS, *ISCAS 96,* May 1996, pp. 481–484.
36. Gibson, D., and Purdy, C.N., Extracting behavioral data from physical descriptions of MEMS for simulation, *Analog Integrated Circuits and Signal Processing 20,* 1999, pp. 227–238.
37. Hayt, W.H., Jr. and Kemmerly, J.E., *Engineering Circuit Analysis,* 5th ed., McGraw-Hill, 1993, pp. 88–95.
38. Dewey, A., Hanna, J., Hillman, B., Dussault, H., Fedder, G., Christen, E., Bakalar, K., Carter, H., and Romanowica, B., VHDL-AMS Modeling Considerations and Styles for Composite Systems, Version 2.0, http://www.ee.duke.edu/research/IMPACT/documents/model_g.pdf (accessed July 20, 2001).
39. McCalla, W.J., *Fundamentals of Computer-Aided Circuit Simulation,* Kluwer Academic, 1988.
40. Clark, J.V., Zhou, N., and Pister, K.S.J., Modified nodal analysis for MEMS with multi-energy domains, *International Conference on Modeling and Simulation of Microsystems, Semiconductors, Sensors and Actuators,* San Diego, CA, March 27–29, 2000, pp. 31–34.
41. Stasa, F.L., *Applied Finite Element Analysis for Engineers,* Holt, Rinehart and Winston, 1985.
42. http://www.wolfram.com/products/mathematica (accessed July 20, 2001).
43. http://www.mathworks.com/products/matlab (accessed July 20, 2001).
44. Mehta, A., Design and Control Oriented Approach to the Modeling of Microfluidic System Components, M.S. Thesis, University of Cincinnati, 1999.
45. Swart, N., Nathan, A., Shams, M., and Parameswaran, M., Numerical optimisation of flow-rate microsensors using circuit simulation tools, *Transducers '91,* 1991, pp. 26–29.
46. http://www.ee.duke.edu/research/IMPACT/vhdl-ams/index.html (accessed July 20, 2001).
47. Gibson, D., Carter, H., and Purdy, C., The use of hardware description languages in the development of microelectromechanical systems, *International Journal of Analog Integrated Circuits and Signal Processing,* 28(2), August 2001, pp. 173–180.

48. http://www.vhdl-ams.com/ (accessed July 20, 2001).
49. http://www.ansys.com/action/MEMSinitiative/index.htm (accessed July 20, 2001).
50. http://www.ansys.com/action/pdf/MEMS_WP.pdf (accessed July 20, 2001).
51. http://www.cfdrc.com (accessed July 20, 2001).
52. Pister, K., SUGAR V2.0, http://www-bsac.EECS.Berkeley.edu/~cfm/ mainpage.html (accessed July 20, 2001).
53. http://www.ece.cmu.edu/~mems/projects/memsyn/nodasv1_4/index.shtml (accessed July 20, 2001).
54. http://www2.ece.cmu.edu/~mems/projects/memsyn/nodasv1_4/tutorial.html (accessed July 20, 2001).
55. Jing, Q. and Fedder, G.K., NODAS 1.3-nodal design of actuators and sensors, *IEEE/VIUF International Workshop on Bahavioral Modeling and Simulation,* Orlando, Fla., October 27–28, 1998.
56. http://www.coventor.com/software/coventorware/index.html (accessed July 20, 2001).
57. Senturia, S.D., Simulation and design of microsystems: a 10-year perspective, *Sensors and Actuators A,* 67, 1998, pp. 1–7.
58. www.memscap.com/index2.html (accessed July 20, 2001).
59. http://www.tanner.com/ (accessed July 20, 2001).
60. http://mems.isi.edu (accessed July 20, 2001).
61. http://mems.isi.edu/mems/materials/index.html (accessed July 20, 2001).
62. http://www.memsrus.com (accessed July 20, 2001).
63. http://www.memsrus.com/cronos/svcsmumps.html (accessed July 20, 2001).
64. Petersen, K., Silicon as a mechanical material, *IEEE Proceedings,* 70(5), May 1982, pp. 420–457.
65. http://www.ece.cmu.edu/~mems/projects/memsyn/index.shtml (accessed July 20, 2001).
66. Beyette, F., Jr. and C.N. Purdy, Teaching modules for a class in mechatronics, *European Workshop on Microelectronics Education (EWME2000),* May 2000.
67. Allen, P.E. and Holberg, D.R., *CMOS Analog Circuit Design,* Oxford University Press, 1987, pp. 142–144.
68. Madou, M., *Fundamentals of Microfabrication,* CRC Press, Boca Raton, FL, 1997.
69. Gibson, D. and Purdy, C., The strengths and weaknesses of VHDL-AMS as a tool for MEMS development, white paper, 2000, http://www.ececs.uc.edu/~cpurdy/csl.html/pub.html/weakvhdl. pdf (accessed July 20, 2001).
70. Mukherjee, T. and Fedder, G.K., Hierarchical mixed-domain circuit simulation, synthesis and extraction methodology for MEMS, *Journal of VLSI Signal Processing,* 21, 1999, pp. 233–249.

# 8

# The Physical Basis of Analogies in Physical System Models




Neville Hogan
*Massachusetts Institute of Technology*

Peter C. Breedveld
*University of Twente*


## 8.1 Introduction

One of the fascinating aspects of mechatronic systems is that their function depends on interactions between electrical and mechanical behavior and often magnetic, fluid, thermal, chemical, or other effects as well. At the same time, this can present a challenge as these phenomena are normally associated with different disciplines of engineering and physics. One useful approach to this multidisciplinary or "multi-physics" problem is to establish analogies between behavior in different domains—for example, resonance due to interaction between inertia and elasticity in a mechanical system is analogous to resonance due to interaction between capacitance and inductance in an electrical circuit. Analogies can provide valuable insight about how a design works, identify equivalent ways a particular function might be achieved, and facilitate detailed quantitative analysis. They are especially useful in studying dynamic behavior, which often arises from interactions between domains; for example, even in the absence of elastic effects, a mass moving in a magnetic field may exhibit resonant oscillation. However, there are many ways that analogies may be established and, unfortunately, the most appropriate analogy between electrical circuits, mechanical and fluid systems remains unresolved: is force like current, or is force more like voltage? In this contribution we examine the physical basis of the analogies in common use and how they may be extended beyond mechanical and electrical systems.

## 8.2 History

It is curious that one of the earliest applications of analogies between electrical and mechanical systems was to enable the demonstration and study of transients in electrical networks that were otherwise too fast to be observed by the instrumentation of the day by identifying mechanical systems with equivalent dynamic behavior; that was the topic of a series of articles on "Models and analogies for demonstrating electrical principles" (*The Engineer,* 1926). Improved methods capable of observing fast electrical transients directly (especially the cathode ray oscilloscope, still in use today) rendered this approach obsolete but enabled quantitative study of nonelectrical systems via analogous electrical circuits (Nickle, 1925). Although that method had considerably more practical importance at the time than it has today (we now have the luxury of vastly more powerful tools for numerical computation of electromechanical system responses), in the late '20s and early '30s a series of papers (Darrieus, 1929; Hähnle, 1932; Firestone, 1933) formulated a rational method to use electrical networks as a framework for establishing analogies between physical systems.

## 8.3 The Force-Current Analogy: Across and through Variables

Firestone identified two types of variable in each physical domain—"across" and "through" variables—which could be distinguished based on how they were measured. An "across" variable may be measured as a difference between values at two points in space (conceptually, *across* two points); a "through" variable may be measured by a sensor in the path of power transmission between two points in space (conceptually, it is transmitted *through* the sensor). By this classification, electrical voltage is analogous to mechanical velocity and electrical current is analogous to mechanical force. Of course, this classification of variables implies a classification of network elements: a mass is analogous to a capacitor, a spring is analogous to an inductor and so forth.

The "force-is-like-current" or "mass-capacitor" analogy has a sound mathematical foundation. Kirchhoff's node law or current law, introduced in 1847 (the sum of currents into a circuit node is identically zero) can be seen as formally analogous to D'Alembert's principle, introduced in 1742 (the sum of forces on a body is identically zero, provided the sum includes the so-called "inertia force," the negative of the body mass times its acceleration). It is the analogy used in linear-graph representations of lumped-parameter systems, proposed by Trent in 1955. Linear graphs bring powerful results from mathematical graph theory to bear on the analysis of lumped-parameter systems. For example, there is a systematic procedure based on partitioning a graph into its *tree* and *links* for selecting sets of independent variables to describe a system. Graph-theoretic approaches are closely related to matrix methods that in turn facilitate computer-aided methods. Linear graphs provide a unified representation of lumped-parameter dynamic behavior in several domains that has been expounded in a number of successful textbooks (e.g., Shearer et al., 1967; Rowell & Wormley, 1997).

The mass-capacitor analogy also appears to afford some practical convenience. It is generally easier to identify points of common velocity in a mechanical system than to identify which elements experience the same force; and it is correspondingly easier to identify the nodes in an electrical circuit than all of its loops. Hence with this analogy it is straightforward to identify an electrical network equivalent to a mechanical system, at least in the one-dimensional case.

### Drawbacks of the Across-Through Classification

Despite the obvious appeal of establishing analogies based on practical measurement procedures, the force-current analogy has some drawbacks that will be reviewed below: (i) on closer examination, measurement-based classification is ambiguous; (ii) its extension to more than one-dimensional mechanical systems is problematical; and (iii) perhaps most important, it leads to analogies (especially between mechanical and fluid systems) that defy common physical insight.

## Measurement as a Basis for Analogies

Even a cursory review of state-of-the-art measurement technologies shows that the across-through classification may be an anachronism or, at best, an over-simplification. Velocity (an "across" variable) may be measured by an integrating accelerometer that is attached only to the point where velocity is measured—that's how the human inner ear measures head velocity. While the velocity is measured with respect to an inertial reference frame (as it should be), there is no tangible connection to that frame. As a further example, current in a conductor (a "through" variable) may be measured without inserting an ammeter in the current path; sensors that measure current by responding to the magnetic field next to the conductor are commercially available (and preferred in some applications). Moreover, in some cases similar methods can be applied to measure both "across" and "through" variables. For example, fluid flow rate is classified as a through variable, presumably by reference to its measurement by, for example, a positive-displacement meter in the flow conduit; that's the kind of fluid measurement commonly used in a household water meter. However, optical methods that are used to measure the velocity of a rigid body (classified as an across variable) are often adapted to measure the volumetric flow rate of a fluid (laser doppler velocimetry is a notable example). Apparently the same fundamental measurement technology can be associated with an across variable in one domain and a through variable in another. Thus, on closer inspection, the definition of across and through variables based on measurement procedures is, at best, ambiguous.

## Beyond One-Dimensional Mechanical Systems

The apparent convenience of equating velocities in a mechanical system with voltages at circuit nodes diminishes rapidly as we go beyond translation in one dimension or rotation about a fixed axis. A translating body may have two or three independent velocities (in planar and spatial motion, respectively). Each independent velocity would appear to require a separate independent circuit node, but the kinetic energy associated with translation can be redistributed at will among these two or three degrees of freedom (e.g., during motion in a circle at constant speed the total kinetic energy remains constant while that associated with each degree of freedom varies). This requires some form of connection between the corresponding circuit nodes in an equivalent electrical network, but what that connection should be is not obvious.

The problem is further exacerbated when we consider rotation. Even the simple case of planar motion (i.e., a body that may rotate while translating) requires three independent velocities, hence three independent nodes in an equivalent electrical network. Reasoning as above we see that these three nodes must be connected but in a different manner from the connection between three nodes equivalent to spatial translation. Again, this connection is hardly obvious, yet translating while rotating is ubiquitous in mechanical systems—that's what a wheel usually does.

Full spatial rotation is still more daunting. In this case interaction between the independent degrees of freedom is especially important as it gives rise to gyroscopic effects, including oscillatory precession and nutation. These phenomena are important practical considerations in modern mechatronics, not arcane subtleties of classical mechanics; for example, they are the fundamental physics underlying several designs for a microelectromechanical (MEMS) vibratory rate gyroscope (Yazdi et al., 1998).

## Physical Intuition

In our view the most important drawback of the across-through classification is that it identifies force as analogous to fluid flow rate as well as electrical current (with velocity analogous to fluid pressure as well as voltage). This is highly counter-intuitive and quite confusing. By this analogy, fluid pressure is not analogous to force despite the fact that pressure is commonly defined as force per unit area. Furthermore, stored kinetic energy due to fluid motion is not analogous to stored kinetic energy due to motion of a rigid body. Given the remarkable similarity of the physical processes underlying these two forms of energy storage, it is hard to understand why they should not be analogous.

Insight is the ultimate goal of modeling. It is a crucial factor in producing innovative and effective designs and depends on developing and maintaining a "physical intuition" about the way devices behave. It is important that analogies between physical effects in different domains can be reconciled with the physical intuition and any method that requires a counter-intuitive analogy is questionable; at a minimum it warrants careful consideration.

## 8.4 Maxwell's Force-Voltage Analogy: Effort and Flow Variables

An alternative analogy classifies variables in each physical domain that (loosely speaking) describe motion or cause it. Thus fluid flow rate, electrical current, and velocity are considered analogous (sometimes generically described as "flow" variables). Conversely, fluid pressure, electrical voltage, and force are considered analogous (sometimes generically described as "effort" variables).

The "force-is-like-voltage" analogy is the oldest drawn between mechanical and electrical systems. It was first proposed by Maxwell (1873) in his treatise on electricity and magnetism, where he observed the similarity between the Lagrangian equations of classical mechanics and electromechanics. That was why Firestone (1933) presented his perspective that force is like current as "A *new* analogy between mechanical and electrical systems" (emphasis added). Probably because of its age, the force-voltage analogy is deeply embedded in our language. In fact, voltage is still referred to as "electromotive force" in some contexts. Words like "resist" or "impede" also have this connotation: a large resistance or impedance implies a large force for a given motion or a large voltage for a given current.

In fact, Maxwell's classification of velocity as analogous to electrical current (with force analogous to voltage) has a deeper justification than the similarity of one mathematical form of the equations of mechanics and electromechanics; it can be traced to a similarity of the underlying physical processes.

### Systems of Particles

Our models of the physical world are commonly introduced by describing systems of particles distributed in space. The particles may have properties such as mass, charge, etc., though in a given context we will deliberately choose to neglect most of those properties so that we may concentrate on a single physical phenomenon of interest. Thus, to describe electrical capacitance, we consider only charge, while to describe translational inertia, we consider only mass and so forth.

Given that this common conceptual model is used in different domains, it may be used to draw analogies between the variables of different physical domains. From this perspective, quantities associated with the motion of particles may be considered analogous to one another; thus mechanical velocity, electrical current, and fluid flow rate are analogous. Accordingly, mechanical displacement, displaced fluid volume, and displaced charge are analogous; and thus force, fluid pressure, and voltage are analogous. This classification of variables obviously implies a classification of network elements: a spring relates mechanical displacement and force; a capacitor relates displaced charge and voltage. Thus a spring is analogous to a capacitor, a mass to an inductor, and for this reason, this analogy is sometimes termed the "mass-inductor" analogy.

### Physical Intuition

The "system-of-particles" models naturally lead to the "intuitive" analogy between pressure, force, and voltage. But, is such a vague and ill-defined concept as "physical intuition" an appropriate consideration in drawing analogies between physical systems? After all, physical intuition might largely be a matter of usage and familiarity, rooted in early educational and cultural background.

We think not; instead we speculate that physical intuition may be related to conformity with a mental model of the physical world. That mental model is important for thinking about physical systems and, if shared, for communicating about them. Because the "system-of-particles" model is widely assumed

(sometimes explicitly, sometimes implicitly) in the textbooks and handbooks of basic science and engineering we speculate that it may account for the physical intuition shared by most engineers. If so, then conforming with that common "system-of-particles" mental model is important to facilitate designing, thinking, and communicating about mechatronic systems. The force-voltage analogy does so; the force-current analogy does not.

### Dependence on Reference Frames

The "system-of-particles" model also leads to another important physical consideration in the choice of analogies between variables: the way they depend on reference frames. The mechanical displacement that determines the elastic potential energy stored in a spring and the displaced charge that determines the electrostatic potential energy stored in a capacitor may be defined with respect to *any* reference frame (whether time-varying or stationary). In contrast, the motion required for kinetic energy storage in a rigid body or a fluid must be defined with respect to an *inertial* frame. Though it may often be overlooked, the motion of charges required for magnetic field storage must also be defined with respect to an inertial frame (Feynman et al., 1963).

To be more precise, the constitutive equations of energy storage based on motion (e.g., in a mass or an inductor) require an inertial reference frame (or must be modified in a non-inertial reference frame). In contrast, the constitutive equations of energy storage based on displacement (e.g., in a spring or a capacitor) do not. Therefore, the mass-inductor (force-voltage) analogy is more consistent with fundamental physics than the mass-capacitor (force-current) analogy.

The modification of the constitutive equations for magnetic energy storage in a non-inertial reference frame is related to the transmission of electromagnetic radiation. However, Kirchhoff's laws (more aptly termed "Kirchhoff's approximations"), which are the foundations of electric network theory, are equivalent to assuming that electromagnetic radiation is absent or negligible. It might, therefore, be argued that the dependence of magnetic energy storage on an inertial reference frame is negligible for electrical circuits, and hence is irrelevant for any discussion of the physical basis of analogies between electrical circuits and other lumped-parameter dynamic-system models. That is undeniably true and could be used to justify the force-current analogy. Nevertheless, because of the confusion that can ensue, the value of an analogy that is fundamentally inconsistent with the underlying physics of lumped-parameter models is questionable.

## 8.5 A Thermodynamic Basis for Analogies

Often in the design and analysis of mechatronic systems it is necessary to consider a broader suite of phenomena than those of mechanics and electromechanics. For instance, it may be important to consider thermal conduction, convection, or even chemical reactions and more. To draw analogies between the variables of these domains it is helpful to examine the underlying physics. The analogous dynamic behavior observed in different physical domains (resonant oscillation, relaxation to equilibrium, etc.) is not merely a similarity of *mathematical* forms, it has a common *physical* basis which lies in the storage, transmission, and irreversible dissipation of energy. Consideration of energy leads us to thermodynamics; we show next that thermodynamics provides a broader basis for drawing analogies and yields some additional insight.

All of the displacements considered to be analogous above (i.e., mechanical displacement, displaced fluid volume, and displaced charge) may be associated with an energy storage function that requires equilibrium for its definition, the displacement being the argument of that energy function. Generically, these may be termed potential energy functions. To elaborate, elastic energy storage requires sustained but recoverable deformation of a material (e.g., as in a spring); the force required to sustain that deformation is determined at equilibrium, defined when the time rate of change of relative displacement of the material particles is uniformly zero (i.e., all the particles are at rest relative to each other). Electrostatic energy storage requires sustained separation of mobile charges of opposite sign (e.g., as in

a capacitor); the required voltage is determined at equilibrium, defined when the time rate of change of charge motion is zero (i.e., all the charges are at rest relative to each other).

## Extensive and Intensive Variables

In the formalism of thermodynamics, the amount of stored energy and the displacement that determines it are *extensive* variables. That is, they vary with the spatial extent (i.e., size or volume) of the object storing the energy. The total elastic energy stored in a uniform rod of constant cross-sectional area in an idealized uniform state of stress is proportional to the length (and hence volume) of the rod; so is the total relative displacement of its ends; both are extensive variables. The total electrostatic energy stored in an idealized parallel-plate capacitor (i.e., one with no fringe fields) is proportional to the area of the plates (and hence, for constant gap, the volume they enclose); so is the total separated charge on the plates; both are extensive variables (cf., Breedveld, 1984).

Equilibrium of these storage elements is established by an *intensive* variable that does not change with the size of the object. This variable is the gradient (partial derivative) of the stored energy with respect to the corresponding displacement. Thus, at equilibrium, the force on each cross-section of the rod is the same regardless of the length or volume of the rod; force is an intensive variable. If the total charge separated in the capacitor is proportional to area, the voltage across the plates is independent of area; voltage is an intensive variable.

Dynamics is not solely due to the storage of energy but arises from the transmission and deployment of power. The instantaneous power into an equilibrium storage element is the product of the (intensive) gradient variable (force, voltage) with the time rate of change of the (extensive) displacement variable (velocity, current). Using this thermodynamics-based approach, all intensive variables are considered analogous, as are all extensive variables and their time rates of change, and so on.

Drawing analogies from a thermodynamic classification into extensive and intensive variables may readily be applied to fluid systems. Consider the potential energy stored in an open container of incompressible fluid: The pressure at any specified depth is independent of the area at that depth and the volume of fluid above it; pressure is an intensive variable analogous to force and voltage, as our common physical intuition suggests it should be. Conversely, the energy stored in the fluid above that depth is determined by the volume of fluid; energy and volume are extensive variables, volume playing the role of displacement analogous to electrical charge and mechanical displacement. Pressure is the partial derivative of stored energy with respect to volume and the instantaneous power into storage is the product of pressure with volumetric flow rate, the time rate of change of volume flowing past the specified depth.

An important advantage of drawing analogies from a classification into extensive and intensive variables is that it may readily be generalized to domains to which the "system-of-particles" image may be less applicable. For example, most mechatronic designs require careful consideration of heating and cooling but there is no obvious flow of particles associated with heat flux. Nevertheless, extensive and intensive variables associated with equilibrium thermal energy storage can readily be identified. Drawing on classical thermodynamics, it can be seen that (total) entropy is an extensive variable and plays the role of a displacement. The gradient of energy with respect to energy is temperature, an intensive variable, which should be considered analogous to force, voltage, and pressure. Equality of temperature establishes thermal equilibrium between two bodies that may store heat (energy) and communicate it to one another.

A word of caution is appropriate here as a classification into extensive and intensive variables properly applies only to scalar quantities such as pressure, volume, etc. As outlined below, the classification can be generalized in a rigorous way to nonscalar quantities, but care is required (cf., Breedveld, 1984).

## Equilibrium and Steady State

In some (though not all) domains energy storage may also be based on motion. Kinetic energy storage may be associated with rigid body motion or fluid motion; magnetic energy storage requires motion of charges. The thermodynamics-based classification properly groups these different kinds of energy storage as analogous to one another and generically they may be termed *kinetic* energy storage elements.

All of the motion variables considered to be analogous (i.e., velocity, fluid flow rate, current) may be associated with an energy storage function that is defined by *steady state* (rather than by equilibrium). For a rigid body, steady motion requires zero net force, and hence constant momentum and kinetic energy. For the magnetic field that stores energy in an inductor, steady current requires zero voltage, and hence constant magnetic flux and magnetic energy.

It might reasonably be argued that any distinction between equilibrium and steady state is purely a matter of perspective and common usage, rather than a fundamental feature of the physical world. For example, with an alternative choice of reference frames, "sustained motion" could be redefined as "rest" or "equilibrium." From this perspective, a zero-relative-velocity "equilibrium" between two rigid bodies (or between a rigid body and a reference frame) could be defined by zero force. Following this line of reasoning any distinction between the mass-inductor and mass-capacitor analogies would appear to be purely a matter of personal choice. However, while the apparent equivalence of "equilibrium" and "steady state" may be justifiable in the formal mathematical sense of zero rate of change of a variable, in a mechanical system, displacement (or position) and velocity (or momentum) are fundamentally different. For example, whereas velocity, force, and momentum may be transformed between reference frames as rank-one tensors, position (or displacement) may not be transformed as a tensor of any kind. Thus, a distinction between equilibrium and steady state reflects an important aspect of the structure of physical system models.

## Analogies, Not Identities

It is important to remember that any classification to establish analogies is an abstraction. At most, dynamic behavior in different domains may be similar; it is not identical. We have pointed out above that if velocity or current is used as the argument of an energy storage function, care must be taken to identify an appropriate inertial reference frame and/or to understand the consequences of using a non-inertial frame. However, another important feature of these variables is that they are fundamentally vectors (i.e., they have a definable spatial orientation). One consequence is that the thermodynamic definition of extensive and intensive variables must be generalized before it may be used to classify these variables (cf., Breedveld, 1984). In contrast, a quantity such as temperature or pressure is fundamentally a scalar. Furthermore, both of these quantities are intrinsically "positive" scalars insofar as they have well-defined, unique and physically meaningful zero values (absolute zero temperature, the pressure of a perfect vacuum). Quite aside from any dependence on inertial reference frames, the across-through analogy between velocity (a vector with no unique zero value) and pressure (a scalar with a physically important zero) will cause error and confusion if used without due care.

This consideration becomes especially important when similar elements of a model are combined (for example, a number of bodies moving with identical velocity may be treated as a single rigid body) to simplify the expression of dynamic equations or improve their computability. The engineering variables used to describe energy storage can be categorized into two groups: (i) positive-valued scalar variables and (ii) nonscalar variables. Positive-valued scalar variables have a physically meaningful zero or absolute reference; examples include the volume of stored fluid, the number of moles of a chemical species, entropy, etc. Nonscalar[1] variables have a definable spatial orientation. Even in the one-dimensional case they can be positive or negative, the sign denoting direction with respect to some reference frame; examples include displacement, momentum, etc. These variables generally do not have a physically meaningful zero or absolute reference, though some of them must be defined with respect to an inertial frame.

Elements of a model that describe energy storage based on scalar variables can be combined in only one way: they must be in mutual equilibrium; their extensive variables are added, while the corresponding intensive variables are equal, independent of direction, and determine the equilibrium condition. For model elements that describe energy storage based on nonscalar variables there are usually two options.

[1]The term "vector variables" suggests itself but these variables may include three-dimensional spatial orientation, which may not be described as a vector.

Electrical capacitors, for instance, may be combined in parallel or in series and the resulting equivalent capacitor may readily be determined. In a parallel connection, equilibrium is determined by voltage (an intensive variable) and the electric charges (extensive variables) are added as before. However, a series connection is the "dual" in the sense that the roles of charge and voltage are exchanged: equality of charges determines equilibrium and the voltages are added. Mechanical springs may also be combined in two ways. However, that is not the case for translational masses and rotational inertias; they may only be combined into a single equivalent rigid body if their velocities are equal and in that case their momenta are added.

The existence of two "dual" ways to combine some, but not all, of the energy storage elements based on nonscalar quantities is somewhat confusing. It may have contributed to the lengthy debate (if we date its beginning to Maxwell, lasting for over a century!) on the best analogy between mechanical and electrical systems. Nevertheless, the important point is that series and parallel connections may not be generalized in a straightforward way to all domains.

### Nodicity

As insight is the foremost goal of modeling, analogies should be chosen to promote insight. Because there may be fundamental differences between all of the physical domains, care should be exercised in drawing analogies to ensure that special properties of one domain should not be applied inappropriately to other domains. This brings us to what may well be the strongest argument against the across-through classification. History suggests that it originated with the use of equivalent electrical network representations of nonelectrical systems. Unfortunately, electrical networks provide an inappropriate basis for developing a general representation of physical system dynamics. This is because electrical networks enjoy a special property, *nodicity*, which is quite unusual among the physical system domains (except as an approximation).

Nodicity refers to the fact that any sub-network (cut-set) of an electrical network behaves as a node in the sense that a Kirchhoff current balance equation may be written for the entire sub-network. As a result of nodicity, electrical network elements can be assembled in arbitrary topologies and yet still describe a physically realizable electrical network. This property of "arbitrary connectability" is not a general property of lumped-parameter physical system models. Most notably, mass elements cannot be connected arbitrarily; they must always be referenced to an inertial frame. For that reason, electrical networks can be quite misleading when used as a basis for a general representation of physical system dynamics. This is not merely a mathematical nicety; some consequences of non-nodic behavior for control system analysis have recently been explored (Won and Hogan, 1998).

By extension, because each of the physical domains has its unique characteristics, any attempt to formulate analogies by taking one of the domains (electrical, mechanical, or otherwise) as a starting point is likely to have limitations. A more productive approach is to begin with those characteristics of physical variables common to all domains and that is the reason to turn to thermodynamics. In other words, the best way to identify analogies *between* domains may be to "step outside" *all* of them. By design, general characteristics of all domains such as the extensive nature of stored energy, the intensive nature of the variables that define equilibrium, and so forth, are not subject to the limitations of any one (such as nodicity). That is the main advantage of drawing analogies based on thermodynamic concepts such as the distinction between extensive and intensive variables.

## 8.6 Graphical Representations

Analogies are often associated with abstract graphical representations of multi-domain physical system models. The force-current analogy is usually associated with the linear graph representation of networks introduced by Trent (1955); the force-voltage analogy is usually associated with the bond graph representation introduced by Paynter (1960). Bond graphs classify variables into efforts (commonly force, voltage, pressure, and so forth) and flows (commonly velocity, current, fluid flow rate, and so forth). Bond graphs extend all the practical benefits of the force-current (across-through) analogy to the force-voltage (effort-flow) analogy: they provide a unified representation of lumped-parameter dynamic behavior in several

domains that has been expounded in a number of successful textbooks (e.g., Karnopp et al., 1975, 1999), there are systematic methods for selecting sets of independent variables to describe a system, ways to take advantage of the ease of identifying velocities and voltages, and matrix methods to facilitate computer analysis. In fact, several computer-aided modeling support packages using the bond-graph language are now available. Furthermore, bond graphs have been applied successfully to describe the dynamics of spatial mechanisms (including gyroscopic effects) while, to the authors' knowledge, linear graphs have not.

Although the force-voltage analogy is most commonly used with bond graphs, the force-current analogy can be used just as readily; the underlying mathematical formalism is indifferent to the choice of which variables are chosen as analogous. In fact, pursuing this line of thought, the choice is unnecessary and may be avoided; doing so affords a way to clarify the potential confusion over the role of intensive variables and the dual types of connection available for some elements in some domains.

In the Generalized Bond Graph (GBG) approach (Breedveld, 1984) all energy storage becomes analogous and only one type of storage element, a (generalized) capacitor, is identified. Its displacement is an extensive variable; the gradient of its energy storage function with respect to that displacement is an intensive variable. In some (but not all) domains a particular kind of coupling known as a *gyrator* is found that gives rise to the *appearance* of a dual type of energy storage, a (generalized) inertia as well as the possibility of dual ways to connect elements. The GBG representation emphasizes the point that the presence of dual types of energy storage and dual types of connection is a special property (albeit an important one) of a limited number of domains. In principle, either a "mass-capacitor" analogy or a "mass-inductor" analogy can be derived from a GBG representation by choosing to associate the gyrating coupling with either the "equilibrium" or "steady-state" energy storage elements.

The important point to be taken here is that the basis of analogies between domains does not depend on the use of a particular abstract graphical representation. The practical value of establishing analogies between domains and the merits of a domain-independent approach based on intensive vs. extensive variables remains regardless of which graph-theoretic tools (if any) are used for analysis.

## 8.7 Concluding Remarks

In the foregoing we articulated some important considerations in the choice of analogies between variables in different physical domains. From a strictly mathematical viewpoint there is little to choose; both analogies may be used as a basis for rigorous, self-consistent descriptions of physical systems. The substantive and important factors emerge from a physical viewpoint—considering the structured way physical behavior is described in the different domains. Summarizing:

- The "system-of-particles" model that is widely assumed in basic science and engineering naturally leads to the intuitive analogy between force and voltage, velocity and current, a mass and an inductor, and so on.
- The measurement procedures used to motivate the distinction between across and through variables at best yield an ambiguous classification.
- Nodicity (the property of "arbitrary connectability") is not a general property of lumped-parameter physical system models. Thus, electrical networks, which are nodic, can be quite misleading when used as a basis for a general representation of physical system dynamics.
- The intuitive analogy between velocity and current is consistent with a thermodynamic classification into extensive and intensive variables. As a result, the analogy can be generalized to dynamic behavior in domains to which the "system-of-particles" image may be less applicable.
- The force-voltage or mass-inductor analogy reflects an important distinction between equilibrium energy-storage phenomena and steady-state energy-storage phenomena: the constitutive equations of steady-state energy storage phenomena require an inertial reference frame (or must be modified in a non-inertial reference frame) while the constitutive equations of equilibrium energy storage phenomena do not.

Our reasoning is based on an assumption that models of physical system dynamics should properly reflect the way descriptions of physical phenomena depend on reference frames and should be compatible with thermodynamics. The across-through classification of variables does not meet these requirements. By contrast, the classification of variables based on the system-of-particles point of view that leads to an analogy between force, pressure, and voltage on the one hand and velocity, fluid flow, and current on the other not only satisfies these criteria, but is the least artificial from a common-sense point of view. We believe this facilitates communication and promotes insight, which are the ultimate benefits of using analogies.

## Acknowledgments

Neville Hogan was supported in part by grant number AR40029 from the National Institutes of Health.

## References

(1926). Models and analogies for demonstrating electrical principles, parts I-XIX. *The Engineer,* 142.

Breedveld, P.C. (1984). *Physical Systems Theory in Terms of Bond Graphs,* University of Twente, Enschede, Netherlands, ISBN 90-9000599-4 (distr. by author).

Darrieus, M. (1929). Les modeles mecaniques en electrotechnique. Leur application aux problemes de stabilite. *Bull. Soc. Franc. Electric.,* 36:729–809.

Feynman, R.P., Leighton, R.B., and Sands, M. (1963). *The Feynman Lectures on Physics, Volume II: Mainly Electromagnetism and Matter,* Addison-Wesley Publishing Company.

Firestone, F.A. (1933). A new analogy between mechanical and electrical system elements. *Journal of the Acoustic Society of America,* 3:249–267.

Hähnle, W. (1932). Die darstellung elektromechanischer gebilde durch rein elektrsiche schaltbilder. *Wissenschaftliche Veroffentl. Siemens Konzern,* 11:1–23.

Karnopp, D.C. and Rosenberg, R.C. (1975). *System Dynamics: A Unified Approach,* John Wiley.

Karnopp, D.C., Margolis, D.L., and Rosenberg, R.C. (1999). *System Dynamics: Modeling and Simulation of Mechatronic Systems,* 3rd edition, John Wiley.

Maxwell, J.C. (1873). *Treatise on Electricity and Magnetism*.

Nickle, C.A. (1925). Oscillographic solutions of electro-mechanical systems. *Trans. A.I.E.E.,* 44:844–856.

Rowell, D. and Wormley, D.N. (1997). *System Dynamics: An Introduction,* Prentice-Hall, NJ.

Shearer, J.L., Murphy, A.T., and Richardson, H.H. (1967). *Introduction to System Dynamics,* Addison-Wesley Publishing Company.

Trent, H.M. (1955). Isomorphisms between oriented linear graphs and lumped physical systems. *Journal of the Acoustic Society of America,* 27:500–527.

Won, J. and Hogan, N. (1998). Coupled stability of non-nodic physical systems. *IFAC Symposium on Nonlinear Control Systems Design.*

Yazdi, N., Ayazi, F., and Najafi, K. (1998). Micromachined inertial sensors. *Proc. IEEE,* 86(8), 1640–1659.

# 9

# Introduction to Sensors and Actuators

M. Anjanappa
*University of Maryland Baltimore County*

K. Datta
*University of Maryland Baltimore County*

T. Song
*University of Maryland Baltimore County*



Sensors and actuators are two critical components of every closed loop control system. Such a system is also called a *mechatronics system*. A typical mechatronics system as shown in Figure 9.1 consists of a sensing unit, a controller, and an actuating unit. A sensing unit can be as simple as a single sensor or can consist of additional components such as filters, amplifiers, modulators, and other signal conditioners. The controller accepts the information from the sensing unit, makes decisions based on the control algorithm, and outputs commands to the actuating unit. The actuating unit consists of an actuator and optionally a power supply and a coupling mechanism.

## 9.1 Sensors

Sensor is a device that when exposed to a physical phenomenon (temperature, displacement, force, etc.) produces a proportional output signal (electrical, mechanical, magnetic, etc.). The term transducer is often used synonymously with sensors. However, ideally, a sensor is a device that responds to a change in the physical phenomenon. On the other hand, a transducer is a device that converts one form of energy into another form of energy. Sensors are transducers when they sense one form of energy input and output in a different form of energy. For example, a thermocouple responds to a temperature change (thermal energy) and outputs a proportional change in electromotive force (electrical energy). Therefore, a thermocouple can be called a sensor and or transducer.

### Classification

Table 9.1 lists various types of sensors that are classified by their measurement objectives. Although this list is by no means exhaustive, it covers all the basic types including the new generation sensors such as smart material sensors, microsensors, and nanosensors.

**TABLE 9.1** Type of Sensors for Various Measurement Objectives

| Sensor | Features |
|---|---|
| | Linear/Rotational sensors |
| Linear/Rotational variable differential transducer (LVDT/RVDT) | High resolution with wide range capability<br>Very stable in static and quasi-static applications |
| Optical encoder | Simple, reliable, and low-cost solution<br>Good for both absolute and incremental measurements |
| Electrical tachometer | Resolution depends on type such as generator or magnetic pickups |
| Hall effect sensor | High accuracy over a small to medium range |
| Capacitive transducer | Very high resolution with high sensitivity<br>Low power requirements<br>Good for high frequency dynamic measurements |
| Strain gauge elements | Very high accuracy in small ranges<br>Provides high resolution at low noise levels |
| Interferometer | Laser systems provide extremely high resolution in large ranges<br>Very reliable and expensive |
| Magnetic pickup | Output is sinusoidal |
| Gyroscope | |
| Inductosyn | Very high resolution over small ranges |
| | Acceleration sensors |
| Seismic accelerometer | Good for measuring frequencies up to 40% of its natural frequency |
| Piezoelectric accelerometer | High sensitivity, compact, and rugged<br>Very high natural frequency (100 kHz typical) |
| | Force, torque, and pressure sensor |
| Strain gauge | Good for both static and dynamic measurements |
| Dynamometers/load cells | They are also available as micro- and nanosensors |
| Piezoelectric load cells | Good for high precision dynamic force measurements |
| Tactile sensor | Compact, has wide dynamic range, and high |
| Ultrasonic stress sensor | Good for small force measurements |
| | Flow sensors |
| Pitot tube | Widely used as a flow rate sensor to determine speed in aircrafts |
| Orifice plate | Least expensive with limited range |
| Flow nozzle, venturi tubes | Accurate on wide range of flow<br>More complex and expensive |
| Rotameter | Good for upstream flow measurements<br>Used in conjunction with variable inductance sensor |
| Ultrasonic type | Good for very high flow rates<br>Can be used for both upstream and downstream flow measurements |
| Turbine flow meter | Not suited for fluids containing abrasive particles<br>Relationship between flow rate and angular velocity is linear |
| Electromagnetic flow meter | Least intrusive as it is noncontact type<br>Can be used with fluids that are corrosive, contaminated, etc.<br>The fluid has to be electrically conductive |
| | Temperature sensors |
| Thermocouples | This is the cheapest and the most versatile sensor<br>Applicable over wide temperature ranges (−200°C to 1200°C typical) |
| Thermistors | Very high sensitivity in medium ranges (up to 100°C typical)<br>Compact but nonlinear in nature |
| Thermodiodes, thermo transistors | Ideally suited for chip temperature measurements<br>Minimized self heating |
| RTD—resistance temperature detector | More stable over a long period of time compared to thermocouple<br>Linear over a wide range |

(*continued*)

**TABLE 9.1** Type of Sensors for Various Measurement Objectives (Continued)

| Sensor | Features |
|---|---|
| Infrared type | Noncontact point sensor with resolution limited by wavelength |
| Infrared thermography | Measures whole-field temperature distribution |
| | Proximity sensors |
| Inductance, eddy current, hall effect, photoelectric, capacitance, etc. | Robust noncontact switching action<br>The digital outputs are often directly fed to the digital controller |
| | Light sensors |
| Photoresistors, photodiodes, photo transistors, photo conductors, etc. | Measure light intensity with high sensitivity<br>Inexpensive, reliable, and noncontact sensor |
| Charge-coupled diode | Captures digital image of a field of vision |
| | Smart material sensors |
| Optical fiber | |
| As strain sensor | Alternate to strain gages with very high accuracy and bandwidth<br>Sensitive to the reflecting surface's orientation and status |
| As level sensor | Reliable and accurate |
| As force sensor | High resolution in wide ranges |
| As temperature sensor | High resolution and range (up to 2000°C) |
| Piezoelectric | |
| As strain sensor | Distributed sensing with high resolution and bandwidth |
| As force sensor | Most suitable for dynamic applications |
| As accelerometer | Least hysteresis and good setpoint accuracy |
| Magnetostrictive | |
| As force sensors | Compact force sensor with high resolution and bandwidth<br>Good for distributed and noncontact sensing applications |
| As torque sensor | Accurate, high bandwidth, and noncontact sensor |
| | Micro- and nano-sensors |
| Micro CCD image sensor | Small size, full field image sensor |
| Fiberscope | Small (0.2 mm diameter) field vision scope using SMA coil actuators |
| Micro-ultrasonic sensor | Detects flaws in small pipes |
| Micro-tactile sensor | Detects proximity between the end of catheter and blood vessels |

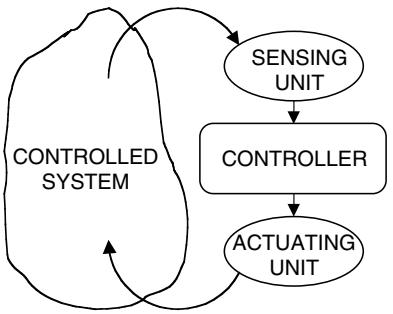



**FIGURE 9.1** A typical mechatronics system.

Sensors can also be classified as *passive* or *active*. In passive sensors, the power required to produce the output is provided by the sensed physical phenomenon itself (such as a thermometer) whereas the active sensors require external power source (such as a strain gage).

Furthermore, sensors are classified as *analog* or *digital* based on the type of output signal. Analog sensors produce continuous signals that are proportional to the sensed parameter and typically require

analog-to-digital conversion before feeding to the digital controller. Digital sensors on the other hand produce digital outputs that can be directly interfaced with the digital controller. Often, the digital outputs are produced by adding an analog-to-digital converter to the sensing unit. If many sensors are required, it is more economical to choose simple analog sensors and interface them to the digital controller equipped with a multi-channel analog-to-digital converter.

## Principle of Operation

### Linear and Rotational Sensors

Linear and rotational position sensors are two of the most fundamental of all measurements used in a typical mechatronics system. The most common type position sensors are listed in Table 9.1. In general, the position sensors produce an electrical output that is proportional to the displacement they experience. There are contact type sensors such as strain gage, LVDT, RVDT, tachometer, etc. The noncontact type includes encoders, hall effect, capacitance, inductance, and interferometer type. They can also be classified based on the range of measurement. Usually the high-resolution type of sensors such as *hall effect, fiber optic inductance, capacitance*, and *strain gage* are suitable for only very small range (typically from 0.1 mm to 5 mm). The *differential transformers* on the other hand, have a much larger range with good resolution. *Interferometer* type sensors provide both very high resolution (in terms of microns) and large range of measurements (typically up to a meter). However, interferometer type sensors are bulky, expensive, and requires large set up time.

Among many linear displacement sensors, strain gage provides high resolution at low noise level and is least expensive. A typical resistance strain gage consists of resistive foil arranged as shown in the Figure 9.2. A typical setup to measure the normal strain of a member loaded in tension is shown in Figure 9.3. Strain gage 1 is bonded to the loading member whereas strain gage 2 is bonded to a second member made of same material, but not loaded. This arrangement compensates for any temperature effect. When the member is loaded, the gage 1 elongates thereby changing the resistance of the gage. The change in resistance is transformed into a change in voltage by the voltage- sensitive wheatstone bridge circuit. Assuming that the resistance of all four arms are equal initially, the change in output voltage ($\Delta v_o$) due to change in resistance ($\Delta R_1$) of gage 1 is

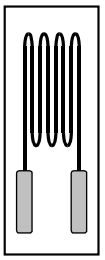


**FIGURE 9.2** Bonded strain gage.

$$\frac{\Delta v_o}{v_i} = \frac{\Delta R_1/R}{4 + 2(\Delta R_1/R)}$$

### Acceleration Sensors

Measurement of acceleration is important for systems subject to shock and vibration. Although acceleration can be derived from the time history data obtainable from linear or rotary sensors, the accelerometers whose output is directly proportional to the acceleration is preferred. Two common types include

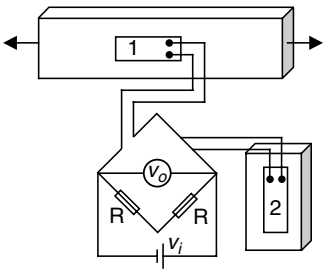



**FIGURE 9.3** Experimental setup to measure normal strain using strain gages.

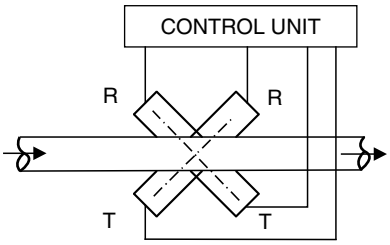



**FIGURE 9.4** Ultrasonic flow sensor arrangement.

the *seismic mass* type and the *piezoelectric* accelerometer. The seismic mass type accelerometer is based on the relative motion between a mass and the supporting structure. The natural frequency of the seismic mass limits its use to low to medium frequency applications. The piezoelectric accelerometer, however, is compact and more suitable for high frequency applications.

### Force, Torque, and Pressure Sensors

Among many type of force/torque sensors, the *strain gage dynamometers* and *piezoelectric type* are most common. Both are available to measure force and/or torque either in one axis or multiple axes. The dynamometers make use of mechanical members that experiences elastic deflection when loaded. These types of sensors are limited by their natural frequency. On the other hand, the piezoelectric sensors are particularly suitable for dynamic loadings in a wide range of frequencies. They provide high stiffness, high resolution over a wide measurement range, and are compact.

### Flow Sensors

Flow sensing is relatively a difficult task. The fluid medium can be liquid, gas, or a mixture of the two. Furthermore, the flow could be laminar or turbulent and can be a time-varying phenomenon. The *venturi meter* and *orifice plate* restrict the flow and use the pressure difference to determine the flow rate. The *pitot tube* pressure probe is another popular method of measuring flow rate. When positioned against the flow, they measure the total and static pressures. The flow velocity and in turn the flow rate can then be determined. The *rotameter* and the *turbine meters* when placed in the flow path, rotate at a speed proportional to the flow rate. The *electromagnetic flow meters* use noncontact method. Magnetic field is applied in the transverse direction of the flow and the fluid acts as the conductor to induce voltage proportional to the flow rate.

*Ultrasonic flow meters* measure fluid velocity by passing high-frequency sound waves through fluid. A schematic diagram of the ultrasonic flow meter is as shown in Figure 9.4. The transmitters (T) provide the sound signal source. As the wave travels towards the receivers (R), its velocity is influenced by the velocity of the fluid flow due to the doppler effect. The control circuit compares the time to interpret the flow rate. This can be used for very high flow rates and can also be used for both upstream and downstream flow. The other advantage is that it can be used for corrosive fluids, fluids with abrasive particles, as it is like a noncontact sensor.

### Temperature Sensors

A variety of devices are available to measure temperature, the most common of which are thermocouples, thermisters, resistance temperature detectors (RTD), and infrared types.

*Thermocouples* are the most versatile, inexpensive, and have a wide range (up to 1200°C typical). A thermocouple simply consists of two dissimilar metal wires joined at the ends to create the sensing junction. When used in conjunction with a reference junction, the temperature difference between the reference junction and the actual temperature shows up as a voltage potential. *Thermisters* are semiconductor devices whose resistance changes as the temperature changes. They are good for very high sensitivity measurements in a limited range of up to 100°C. The relationship between the temperature and the resistance is nonlinear. The *RTD*s use the phenomenon that the resistance of a metal changes with temperature. They are, however, linear over a wide range and most stable.

*Infrared type* sensors use the radiation heat to sense the temperature from a distance. These noncontact sensors can also be used to sense a field of vision to generate a thermal map of a surface.

### Proximity Sensors

They are used to sense the proximity of an object relative to another object. They usually provide a on or off signal indicating the presence or absence of an object. *Inductance, capacitance, photoelectric*, and *hall effect* types are widely used as proximity sensors. Inductance proximity sensors consist of a coil wound around a soft iron core. The inductance of the sensor changes when a ferrous object is in its proximity. This change is converted to a voltage-triggered switch. Capacitance types are similar to inductance except the proximity of an object changes the gap and affects the capacitance. Photoelectric sensors are normally aligned with an infrared light source. The proximity of a moving object interrupts the light beam causing the voltage level to change. Hall effect voltage is produced when a current-carrying conductor is exposed to a transverse magnetic field. The voltage is proportional to transverse distance between the hall effect sensor and an object in its proximity.

### Light Sensors

Light intensity and full field vision are two important measurements used in many control applications. *Phototransistors, photoresistors*, and *photodiodes* are some of the more common type of light intensity sensors. A common photoresistor is made of cadmium sulphide whose resistance is maximum when the sensor is in dark. When the photoresistor is exposed to light, its resistance drops in proportion to the intensity of light. When interfaced with a circuit as shown in Figure 9.5 and balanced, the change in light intensity will show up as change in voltage. These sensors are simple, reliable, and cheap, used widely for measuring light intensity.

### Smart Material Sensors

There are many new smart materials that are gaining more applications as sensors, especially in distributed sensing circumstances. Of these, *optic fibers, piezoelectric,* and *magnetostrictive* materials have found applications. Within these, optic fibers are most used.

Optic fibers can be used to sense strain, liquid level, force, and temperature with very high resolution. Since they are economical for use as *in situ* distributed sensors on large areas, they have found numerous applications in smart structure applications such as damage sensors, vibration sensors, and cure-monitoring sensors. These sensors use the inherent material (glass and silica) property of optical fiber to sense the environment. Figure 9.6 illustrates the basic principle of operation of an embedded optic fiber used to sense displacement, force, or temperature. The relative change in the transmitted intensity or spectrum is proportional to the change in the sensed parameter.

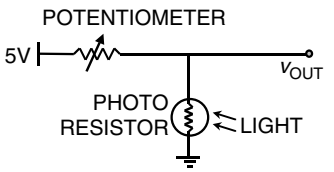



**FIGURE 9.5** Light sensing with photoresistors.

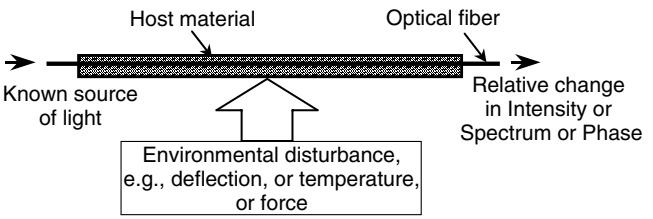



**FIGURE 9.6** Principle of operation of optic fiber sensing.

### Micro- and Nanosensors

Microsensors (sometimes also called MEMS) are the miniaturized version of the conventional macrosensors with improved performance and reduced cost. Silicon micromachining technology has helped the development of many microsensors and continues to be one of the most active research and development topics in this area.

Vision microsensors have found applications in medical technology. A *fiberscope* of approximately 0.2 mm in diameter has been developed to inspect flaws inside tubes. Another example is a *microtactile sensor*, which uses laser light to detect the contact between a catheter and the inner wall of blood vessels during insertion that has sensitivity in the range of 1 mN. Similarly, the progress made in the area of nanotechnology has fuelled the development of nanosensors. These are relatively new sensors that take one step further in the direction of miniaturization and are expected to open new avenues for sensing applications.

## Selection Criteria

A number of static and dynamic factors must be considered in selecting a suitable sensor to measure the desired physical parameter. Following is a list of typical factors:

*Range*—Difference between the maximum and minimum value of the sensed parameter
*Resolution*—The smallest change the sensor can differentiate
*Accuracy*—Difference between the measured value and the true value
*Precision*—Ability to reproduce repeatedly with a given accuracy
*Sensitivity*—Ratio of change in output to a unit change of the input
*Zero offset*—A nonzero value output for no input
*Linearity*—Percentage of deviation from the best-fit linear calibration curve
*Zero Drift*—The departure of output from zero value over a period of time for no input
*Response time*—The time lag between the input and output
*Bandwidth*—Frequency at which the output magnitude drops by 3 dB
*Resonance*—The frequency at which the output magnitude peak occurs
*Operating temperature*—The range in which the sensor performs as specified
*Deadband*—The range of input for which there is no output
*Signal-to-noise ratio*—Ratio between the magnitudes of the signal and the noise at the output

Choosing a sensor that satisfies all the above to the desired specification is difficult, at best. For example, finding a position sensor with micrometer resolution over a range of a meter eliminates most of the sensors. Many times the lack of a cost-effective sensor necessitates redesigning the mechatronic system. It is, therefore, advisable to take a system level approach when selecting a sensor and avoid choosing it in isolation.

Once the above-referred functional factors are satisfied, a short list of sensors can be generated. The final selection will then depend upon the size, extent of signal conditioning, reliability, robustness, maintainability, and cost.

## Signal Conditioning

Normally, the output from a sensor requires post processing of the signals before they can be fed to the controller. The sensor output may have to be demodulated, amplified, filtered, linearized, range quantized, and isolated so that the signal can be accepted by a typical analog-to-digital converter of the controller. Some sensors are available with integrated signal conditioners, such as the microsensors. All the electronics are integrated into one microcircuit and can be directly interfaced with the controllers.

## Calibration

The sensor manufacturer usually provides the calibration curves. If the sensors are stable with no drift, there is no need to recalibrate. However, often the sensor may have to be recalibrated after integrating it with a signal conditioning system. This essentially requires that a known input signal is provided to

the sensor and its output recorded to establish a correct output scale. This process proves the ability to measure reliably and enhances the confidence.

If the sensor is used to measure a time-varying input, dynamic calibration becomes necessary. Use of sinusoidal inputs is the most simple and reliable way of dynamic calibration. However, if generating sinusoidal input becomes impractical (for example, temperature signals) then a step input can substitute for the sinusoidal signal. The transient behavior of step response should yield sufficient information about the dynamic response of the sensor.

## 9.2 Actuators

Actuators are basically the muscle behind a mechatronics system that accepts a control command (mostly in the form of an electrical signal) and produces a change in the physical system by generating force, motion, heat, flow, etc. Normally, the actuators are used in conjunction with the power supply and a coupling mechanism as shown in Figure 9.7. The power unit provides either AC or DC power at the rated voltage and current. The coupling mechanism acts as the interface between the actuator and the physical system. Typical mechanisms include rack and pinion, gear drive, belt drive, lead screw and nut, piston, and linkages.

### Classification

Actuators can be classified based on the type of energy as listed in Table 9.2. The table, although not exhaustive, lists all the basic types. They are essentially of electrical, electromechanical, electromagnetic, hydraulic, or pneumatic type. The new generations of actuators include smart material actuators, micro-actuators, and Nanoactuators.

Actuators can also be classified as *binary* and *continuous* based on the number of stable-state outputs. A relay with two stable states is a good example of a binary actuator. Similarly, a stepper motor is a good example of continuous actuator. When used for a position control, the stepper motor can provide stable outputs with very small incremental motion.

### Principle of Operation

#### Electrical Actuators

Electrical switches are the choice of actuators for most of the on-off type control action. Switching devices such as *diodes, transistors, triacs, MOSFET*, and *relays* accept a low energy level command signal from the controller and switch on or off electrical devices such as motors, valves, and heating elements. For example, a MOSFET switch is shown in Figure 9.8. The gate terminal receives the low energy control signal from the controller that makes or breaks the connection between the power supply and the actuator load. When switches are used, the designer must make sure that *switch bounce* problem is eliminated either by hardware or software.

#### Electromechanical Actuators

The most common electromechanical actuator is a motor that converts electrical energy to mechanical motion. Motors are the principal means of converting electrical energy into mechanical energy in industry. Broadly they can be classified as *DC motors, AC motors*, and *stepper motors*. DC motors operate on DC

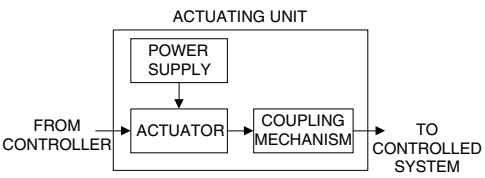



**FIGURE 9.7** A typical actuating unit.

**TABLE 9.2** Type of Actuators and Their Features

| Actuator | | | Features |
|---|---|---|---|
| | | Electrical | |
| Diodes, thyristor, bipolar transistor, triacs, diacs, power MOSFET, solid state relay, etc. | | | Electronic type<br>Very high frequency response<br>Low power consumption |
| | | Electromechanical | |
| DC motor | Wound field | Separately excited | Speed can be controlled either by the voltage across the armature winding or by varying the field current |
| | | Shunt | Constant-speed application |
| | | Series | High starting torque, high acceleration torque, high speed with light load |
| | | Compound | Low starting torque, good speed regulation<br>Instability at heavy loads |
| | Permanent magnet | Conventional PM motor | High efficiency, high peak power, and fast response |
| | | Moving-coil PM motor | Higher efficiency and lower inductance than conventional DC motor |
| | | Torque motor | Designed to run for a long periods in a stalled or a low rpm condition |
| | Electronic commutation (brushless motor) | | Fast response<br>High efficiency, often exceeding 75%<br>Long life, high reliability, no maintenance needed<br>Low radio frequency interference and noise production |
| AC motor | AC induction motor | | The most commonly used motor in industry<br>Simple, rugged, and inexpensive |
| | AC synchronous motor | | Rotor rotates at synchronous speed<br>Very high efficiency over a wide range of speeds and loads<br>Need an additional system to start |
| | Universal motor | | Can operate in DC or AC<br>Very high horsepower per pound ratio<br>Relatively short operating life |
| Stepper motor | Hybrid | | Change electrical pulses into mechanical movement<br>Provide accurate positioning without feedback |
| | Variable reluctance | | Low maintenance |
| | | Electromagnetic | |
| Solenoid type devices<br>Electromagnets, relay | | | Large force, short duration<br>On/off control |
| | | Hydraulic and Pneumatic | |
| Cylinder | | | Suitable for liner movement |
| Hydraulic motor | Gear type | | Wide speed range |
| | Vane type | | High horsepower output |
| | Piston type | | High degree of reliability |
| Air motor | Rotary type | | No electric shock hazard |
| | Reciprocating | | Low maintenance |
| Valves | Directional control valves | | |
| | Pressure control valves | | |
| | Process control valves | | |
| | | Smart Material actuators | |
| Piezoelectric &<br>Electrostrictive | | | High frequency with small motion<br>High voltage with low current excitation<br>High resolution |

(*continued*)

**TABLE 9.2** Type of Actuators and Their Features (Continued)

| Actuator | Features |
|---|---|
| Magnetostrictive | High frequency with small motion<br>Low voltage with high current excitation |
| Shape Memory Alloy | Low voltage with high current excitation<br>Low frequency with large motion |
| Electrorheological fluids | Very high voltage excitation<br>Good resistance to mechanical shock and vibration<br>Low frequency with large force |
| | Micro- and Nanoactuators |
| Micromotors | Suitable for micromechanical system |
| Microvalves | Can use available silicon processing technology, such as electrostatic motor |
| Micropumps | Can use any smart material |

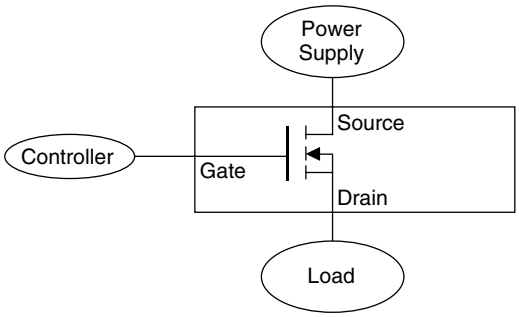



**FIGURE 9.8** n-channel power MOSFET.

voltage and varying the voltage can easily control their speed. They are widely used in applications ranging from thousands of horsepower motors used in rolling mills to fractional horsepower motors used in automobiles (starter motors, fan motors, windshield wiper motors, etc.). Although they are costlier, they need DC power supply and require more maintenance compared to AC motors.

The governing equation of motion of a DC motor can be written as:

$$T = J\frac{d\omega}{dt} + T_L + T_{\text{loss}}$$

where $T$ is torque, $J$ is the total inertia, $\omega$ is the angular mechanical speed of the rotor, $T_L$ is the torque applied to the motor shaft, and $T_{\text{loss}}$ is the internal mechanical losses such as friction.

*AC motors* are the most popular since they use standard AC power, do not require brushes and commutator, and are therefore less expensive. AC motors can be further classified as the *induction motors, synchronous motors*, and *universal motors* according to their physical construction. The induction motor is simple, rugged, and maintenance free. They are available in many sizes and shapes based on number of phases used. For example, a three-phase induction motor is used in large-horsepower applications, such as pump drives, steel mill drives, hoist drives, and vehicle drives. The two-phase servomotor is used extensively in position control systems. Single-phase induction motors are widely used in many household appliances. The synchronous motor is one of the most efficient electrical motors in industry, so it is used in industry to reduce the cost of electrical power. In addition, synchronous motors rotate at synchronous speed, so they are also used in applications that require synchronous operations. The universal motors operate with either

AC or DC power supply. They are normally used in fractional horsepower application. The DC universal motor has the highest horsepower-per-pound ratio, but has a relatively short operating life.

The *stepper motor* is a discrete (incremental) positioning device that moves one step at a time for each pulse command input. Since they accept direct digital commands and produce a mechanical motion, the stepper motors are used widely in industrial control applications. They are mostly used in fractional horsepower applications. With the rapid progress in low cost and high frequency solid-state drives, they are finding increased applications.

Figure 9.9 shows a simplified unipolar stepper motor. The winding-1 is between the top and bottom stator pole, and the winding-2 is between the left and right motor poles. The rotor is a permanent magnet with six poles resulting in a single step angle of 30°. With appropriate excitation of winding-1, the top stator pole becomes a north pole and the bottom stator pole becomes a south pole. This attracts the rotor into the position as shown. Now if the winding-1 is de-energized and winding-2 is energized, the rotor will turn 30°. With appropriate choice of current flow through winding-2, the rotor can be rotated either clockwise or counterclockwise. By exciting the two windings in sequence, the motor can be made to rotate at a desired speed continuously.

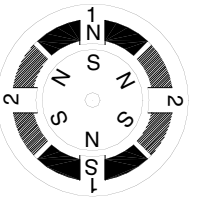



**FIGURE 9.9** Unipolar stepper motor.

## Electromagnetic Actuators

The *solenoid* is the most common electromagnetic actuator. A DC solenoid actuator consists of a soft iron core enclosed within a current carrying coil. When the coil is energized, a magnetic field is established that provides the force to push or pull the iron core. AC solenoid devices are also encountered, such as AC excitation relay.

A solenoid operated directional control valve is shown in Figure 9.10. Normally, due to the spring force, the soft iron core is pushed to the extreme left position as shown. When the solenoid is excited, the soft iron core will move to the right extreme position thus providing the electromagnetic actuation.

Another important type is the *electromagnet*. The electromagnets are used extensively in applications that require large forces.

## Hydraulic and Pneumatic Actuators

Hydraulic and pneumatic actuators are normally either *rotary motors* or *linear piston/cylinder* or *control valves*. They are ideally suited for generating very large forces coupled with large motion. Pneumatic actuators use air under pressure that is most suitable for low to medium force, short stroke, and high-speed applications. Hydraulic actuators use pressurized oil that is incompressible. They can produce very large forces coupled with large motion in a cost-effective manner. The disadvantage with the hydraulic actuators is that they are more complex and need more maintenance.

The rotary motors are usually used in applications where low speed and high torque are required. The cylinder/piston actuators are suited for application of linear motion such as aircraft flap control. Control valves in the form of directional control valves are used in conjunction with rotary motors and cylinders to control the fluid flow direction as shown in Figure 9.10. In this solenoid operated directional control valve, the valve position dictates the direction motion of the cylinder/piston arrangement.

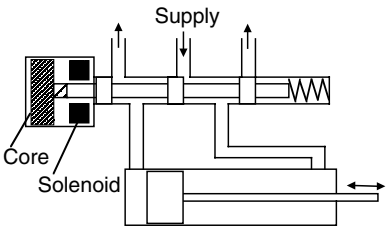



**FIGURE 9.10** Solenoid operated directional control valve.

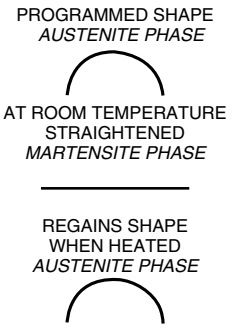



**FIGURE 9.11** Phase changes of Shape Memory Alloy.

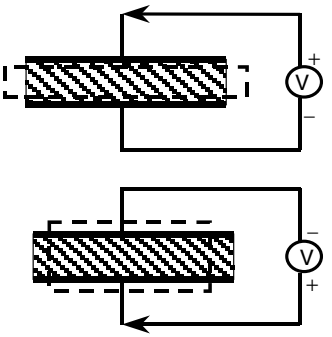



**FIGURE 9.12** Piezoelectric actuator.

### Smart Material Actuators

Unlike the conventional actuators, the smart material actuators typically become part of the load bearing structures. This is achieved by embedding the actuators in a distributed manner and integrating into the load bearing structure that could be used to suppress vibration, cancel the noise, and change shape. Of the many smart material actuators, *shape memory alloys, piezoelectric* (*PZT*), *magnetostrictive, Electrorheological fluids*, and *ion exchange polymers* are most common.

Shape Memory Alloys (SMA) are alloys of nickel and titanium that undergo phase transformation when subjected to a thermal field. The SMAs are also known as NITINOL for Nickel Titanium Naval Ordnance Laboratory. When cooled below a critical temperature, their crystal structure enters martensitic phase as shown in Figure 9.11. In this state the alloy is plastic and can easily be manipulated. When the alloy is heated above the critical temperature (in the range of 50–80°C), the phase changes to austenitic phase. Here the alloy resumes the shape that it formally had at the higher temperature. For example, a straight wire at room temperature can be made to regain its programmed semicircle shape when heated that has found applications in orthodontics and other tensioning devices. The wires are typically heated by passing a current (up to several amperes), 0 at very low voltage (2–10 V typical).

The PZT actuators are essentially piezocrystals with top and bottom conducting films as shown in Figure 9.12. When an electric voltage is applied across the two conducting films, the crystal expands in the transverse direction as shown by the dotted lines. When the voltage polarity is reversed, the crystal contracts thereby providing bidirectional actuation. The interaction between the mechanical and electrical behavior of the piezoelectric materials can be expressed as:

$$T = c^E S - eE$$

where $T$ is the stress, $c^E$ is the elastic coefficients at constant electric field, $S$ is the strain, $e$ is the dielectric permitivity, and $E$ is the electric field.

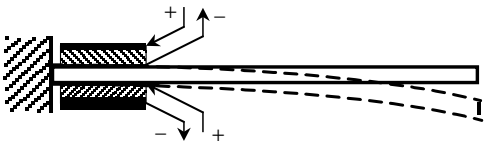



**FIGURE 9.13** Vibration of beam using piezoelectric actuators.

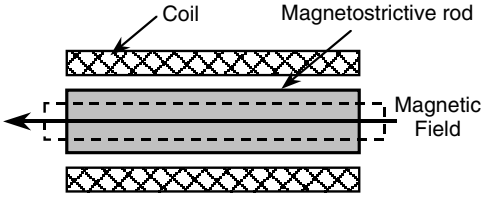



**FIGURE 9.14** Magnetostrictive rod actuator.

One application of these actuators is as shown in Figure 9.13. The two piezoelectric patches are excited with opposite polarity to create transverse vibration in the cantilever beam. These actuators provide high bandwidth (0–10 kHz typical) with small displacement. Since there are no moving parts to the actuator, it is compact and ideally suited for micro and nano actuation. Unlike the bidirectional actuation of piezoelectric actuators, the *electrostriction* effect is a second-order effect, i.e., it responds to an electric field with unidirectional expansion regardless of polarity.

*Magnetostrictive* material is an alloy of terbium, dysprosium, and iron that generates mechanical strains up to 2000 microstrain in response to applied magnetic fields. They are available in the form of rods, plates, washers, and powder. Figure 9.14 shows a typical magnetostrictive rod actuator that is surrounded by a magnetic coil. When the coil is excited, the rod elongates in proportion to the intensity of the magnetic field established. The magnetomechanical relationship is given as:

$$\varepsilon = S^H \sigma + dH$$

where, $\varepsilon$ is the strain, $S^H$ the compliance at constant magnetic filed, $\sigma$ the stress, $d$ the magnetostriction constant, and $H$ the magnetic field intensity.

*Ion exchange polymers* exploit the electro-osmosis phenomenon of the natural ionic polymers for purposes of actuation. When a voltage potential is applied across the cross-linked polyelectrolytic network, the ionizable groups attain a net charge generating a mechanical deformation. These types of actuators have been used to develop artificial muscles and artificial limbs. The primary advantage is their capacity to produce large deformation with a relatively low voltage excitation.

### Micro- and Nanoactuators

Microactuators, also called micromachines, microelectromechanical system (MEMS), and microsystems are the tiny mobile devices being developed utilizing the standard microelectronics processes with the integration of semiconductors and machined micromechanical elements. Another definition states that any device produced by assembling extremely small functional parts of around 1–15 mm is called a micromachine.

In *electrostatic motors*, electrostatic force is dominant, unlike the conventional motors that are based on magnetic forces. For smaller micromechanical systems the electrostatic forces are well suited as an actuating force. Figure 9.15 shows one type of electrostatic motor. The rotor is an annular disk with uniform permitivity and conductivity. In operation, a voltage is applied to the two conducting parallel

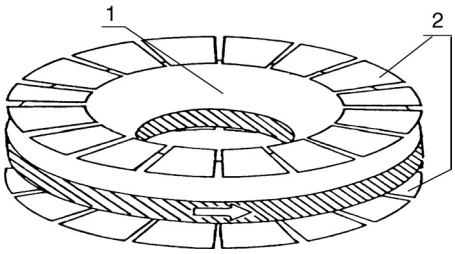



**FIGURE 9.15** Electrostatic motor: 1-rotor, 2-stator electrodes.

plates separated by an insulation layer. The rotor rotates with a constant velocity between the two coplanar concentric arrays of stator electrodes.

## Selection Criteria

The selection of the proper actuator is more complicated than selection of the sensors, primarily due to their effect on the dynamic behavior of the overall system. Furthermore, the selection of the actuator dominates the power needs and the coupling mechanisms of the entire system. The coupling mechanism can sometimes be completely avoided if the actuator provides the output that can be directly interfaced to the physical system. For example, choosing a linear motor in place of a rotary motor can eliminate the need of a coupling mechanism to convert rotary motion to linear motion.

In general, the following performance parameters must be addressed before choosing an actuator for a specific need:

*Continuous power output*—The maximum force/torque attainable continuously without exceeding the temperature limits
*Range of motion*—The range of linear/rotary motion
*Resolution*—The minimum increment of force/torque attainable
*Accuracy*—Linearity of the relationship between the input and output
*Peak force/torque*—The force/torque at which the actuator stalls
*Heat dissipation*—Maximum wattage of heat dissipation in continuous operation
*Speed characteristics*—Force/torque versus speed relationship
*No load speed*—Typical operating speed/velocity with no external load
*Frequency response*—The range of frequency over which the output follows the input faithfully, applicable to linear actuators
*Power requirement*—Type of power (AC or DC), number of phases, voltage level, and current capacity

In addition to the above-referred criteria, many other factors become important depending upon the type of power and the coupling mechanism required. For example, if a rack- and-pinion coupling mechanism is chosen, the backlash and friction will affect the resolution of the actuating unit.

# 10

# Fundamentals of Time and Frequency




Michael A. Lombardi
*National Institute of Standards and Technology*


## 10.1 Introduction

Time and frequency standards supply three basic types of information: *time-of-day, time interval*, and *frequency*. Time-of-day information is provided in hours, minutes, and seconds, but often also includes the *date* (month, day, and year). A device that displays or records time-of-day information is called a *clock*. If a clock is used to label when an event happened, this label is sometimes called a *time tag* or *time stamp*. Date and time-of-day can also be used to ensure that events are *synchronized*, or happen at the same time.

Time interval is the duration or elapsed time between two events. The standard unit of time interval is the second(s). However, many engineering applications require the measurement of shorter time intervals, such as milliseconds (1 ms = $10^{-3}$ s), microseconds (1 $\mu$s = $10^{-6}$ s), nanoseconds (1 ns = $10^{-9}$ s), and picoseconds (1 ps = $10^{-12}$ s). Time is one of the seven base physical quantities, and the second is one of seven base units defined in the International System of Units (SI). The definitions of many other physical quantities rely upon the definition of the second. The second was once defined based on the earth's rotational rate or as a fraction of the tropical year. That changed in 1967 when the era of atomic time keeping formally began. The current definition of the SI second is:

> The duration of 9,192,631,770 periods of the radiation corresponding to the transition between two hyperfine levels of the ground state of the cesium-133 atom.

Frequency is the rate of a repetitive event. If $T$ is the period of a repetitive event, then the frequency $f$ is its reciprocal, $1/T$. Conversely, the period is the reciprocal of the frequency, $T = 1/f$. Since the period is a time interval expressed in seconds (s), it is easy to see the close relationship between time interval and frequency. The standard unit for frequency is the hertz (Hz), defined as events or cycles per second. The frequency of electrical signals is often measured in multiples of hertz, including kilohertz (kHz), megahertz (MHz), or gigahertz (GHz), where 1 kHz equals one thousand ($10^3$) events per second, 1 MHz

**TABLE 10.1** Uncertainties of Physical Realizations of the Base SI Units

| SI Base Unit | Physical Quantity | Uncertainty |
|---|---|---|
| Candela | Luminous intensity | $1 \times 10^{-4}$ |
| Kelvin | Temperature | $3 \times 10^{-7}$ |
| Mole | Amount of substance | $8 \times 10^{-8}$ |
| Ampere | Electric current | $4 \times 10^{-8}$ |
| Kilogram | Mass | $1 \times 10^{-8}$ |
| Meter | Length | $1 \times 10^{-12}$ |
| Second | Time interval | $1 \times 10^{-15}$ |

equals one million ($10^6$) events per second, and 1 GHz equals one billion ($10^9$) events per second. A device that produces frequency is called an *oscillator*. The process of setting multiple oscillators to the same frequency is called *syntonization*.

Of course, the three types of time and frequency information are closely related. As mentioned, the standard unit of time interval is the second. By counting seconds, we can determine the date and the time-of-day. And by counting events or cycles per second, we can measure frequency.

Time interval and frequency can now be measured with less uncertainty and more resolution than any other physical quantity. Today, the best time and frequency standards can realize the SI second with uncertainties of $\cong 1 \times 10^{-15}$. Physical realizations of the other base SI units have much larger uncertainties, as shown in Table 10.1 [1–5].

### Coordinated Universal Time (UTC)

The world's major metrology laboratories routinely measure their time and frequency standards and send the measurement data to the Bureau International des Poids et Measures (BIPM) in Sevres, France. The BIPM averages data collected from more than 200 atomic time and frequency standards located at more than 40 laboratories, including the National Institute of Standards and Technology (NIST). As a result of this averaging, the BIPM generates two time scales, International Atomic Time (TAI), and Coordinated Universal Time (UTC). These time scales realize the SI second as closely as possible.

UTC runs at the same frequency as TAI. However, it differs from TAI by an integral number of seconds. This difference increases when *leap seconds* occur. When necessary, leap seconds are added to UTC on either June 30 or December 31. The purpose of adding leap seconds is to keep atomic time (UTC) within ±0.9 s of an older time scale called UT1, which is based on the rotational rate of the earth. Leap seconds have been added to UTC at a rate of slightly less than once per year, beginning in 1972 [3,5].

Keep in mind that the BIPM maintains TAI and UTC as "paper" time scales. The major metrology laboratories use the published data from the BIPM to steer their clocks and oscillators and generate real-time versions of UTC. Many of these laboratories distribute their versions of UTC via radio signals, which are discussed in section 10.4.

You can think of UTC as the ultimate standard for time-of-day, time interval, and frequency. Clocks synchronized to UTC display the same hour, minute, and second all over the world (and remain within one second of UT1). Oscillators syntonized to UTC generate signals that serve as reference standards for time interval and frequency.

## 10.2 Time and Frequency Measurement

Time and frequency measurements follow the conventions used in other areas of metrology. The frequency standard or clock being measured is called the *device under test* (*DUT*). A measurement compares the DUT to a *standard* or *reference*. The standard should outperform the DUT by a specified ratio, called the *test uncertainty ratio* (*TUR*). Ideally, the TUR should be 10:1 or higher. The higher the ratio, the less averaging is required to get valid measurement results.

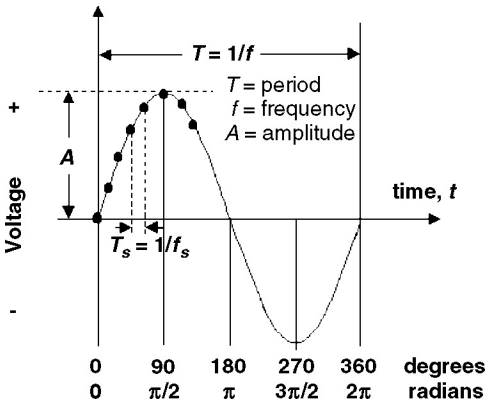



**FIGURE 10.1** An oscillating sine wave.

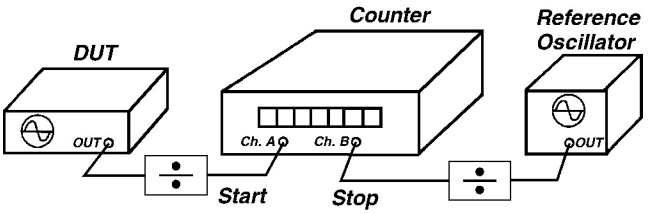



**FIGURE 10.2** Measurement using a time interval counter.

The test signal for time measurements is usually a pulse that occurs once per second (1 pps). The pulse width and polarity varies from device to device, but TTL levels are commonly used. The test signal for frequency measurements is usually at a frequency of 1 MHz or higher, with 5 or 10 MHz being common. Frequency signals are usually sine waves, but can also be pulses or square waves. If the frequency signal is an oscillating sine wave, it might look like the one shown in Figure 10.1. This signal produces one cycle (360° or $2\pi$ radians of phase) in one period. The signal amplitude is expressed in volts, and must be compatible with the measuring instrument. If the amplitude is too small, it might not be able to drive the measuring instrument. If the amplitude is too large, the signal must be attenuated to prevent overdriving the measuring instrument.

This section examines the two main specifications of time and frequency measurements—*accuracy* and *stability*. It also discusses some instruments used to measure time and frequency.

## Accuracy

Accuracy is the degree of conformity of a measured or calculated value to its definition. Accuracy is related to the offset from an ideal value. For example, *time offset* is the difference between a measured on-time pulse and an ideal on-time pulse that coincides exactly with UTC. *Frequency offset* is the difference between a measured frequency and an ideal frequency with zero uncertainty. This ideal frequency is called the *nominal frequency*.

Time offset is usually measured with a *time interval counter* (*TIC*), as shown in Figure 10.2. A TIC has inputs for two signals. One signal starts the counter and the other signal stops it. The time interval between the start and stop signals is measured by counting cycles from the time base oscillator. The resolution of a low cost TIC is limited to the period of its time base. For example, a TIC with a 10-MHz time base oscillator would have a resolution of 100 ns. More elaborate TICs use interpolation schemes to detect parts of a time base cycle and have much higher resolution—1 ns resolution is commonplace, and 20 ps resolution is available.

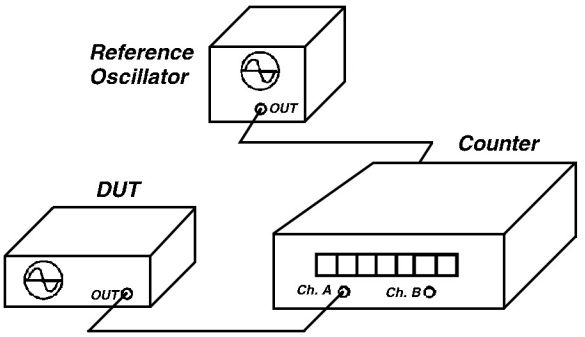



**FIGURE 10.3** Measurement using a frequency counter.

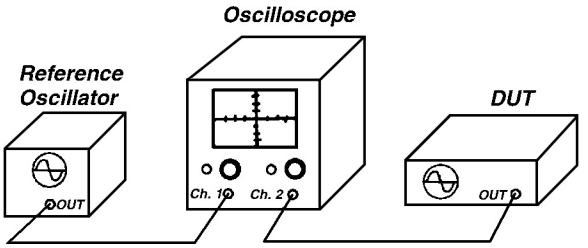



**FIGURE 10.4** Phase comparison using an oscilloscope.

Frequency offset can be measured in either the *frequency domain* or *time domain*. A simple frequency domain measurement involves directly counting and displaying the frequency output of the DUT with a *frequency counter*. The reference for this measurement is either the counter's internal time base oscillator, or an external time base (Figure 10.3). The counter's resolution, or the number of digits it can display, limits its ability to measure frequency offset. For example, a 9-digit frequency counter can detect a frequency offset no smaller than 0.1 Hz at 10 MHz ($1 \times 10^{-8}$). The frequency offset is determined as

$$f(\text{offset}) = \frac{f_{\text{measured}} - f_{\text{nominal}}}{f_{\text{nominal}}}$$

where $f_{\text{measured}}$ is the reading from the frequency counter, and $f_{\text{nominal}}$ is the frequency labeled on the oscillator's nameplate, or specified output frequency.

Frequency offset measurements in the time domain involve a *phase comparison* between the DUT and the reference. A simple phase comparison can be made with an oscilloscope (Figure 10.4). The oscilloscope will display two sine waves (Figure 10.5). The top sine wave represents a signal from the DUT, and the bottom sine wave represents a signal from the reference. If the two frequencies were exactly the same, their phase relationship would not change and both would appear to be stationary on the oscilloscope display. Since the two frequencies are not exactly the same, the reference appears to be stationary and the DUT signal moves. By measuring the rate of motion of the DUT signal we can determine its frequency offset. Vertical lines have been drawn through the points where each sine wave passes through zero. The bottom of the figure shows bars whose width represents the phase difference between the signals. In this case the phase difference is increasing, indicating that the DUT is lower in frequency than the reference.

Measuring high accuracy signals with an oscilloscope is impractical, since the phase relationship between signals changes very slowly and the resolution of the oscilloscope display is limited. More precise phase comparisons can be made with a TIC, using a setup similar to Figure 10.2. If the two input signals have the same frequency, the time interval will not change. If the two signals have different frequencies, the time interval will change, and the rate of change is the frequency offset. The resolution of a TIC

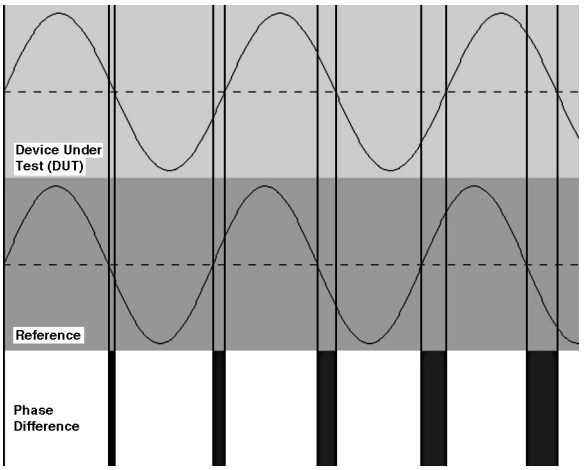



**FIGURE 10.5** Two sine waves with a changing phase relationship.

determines the smallest frequency change that it can detect without averaging. For example, a low cost TIC with a single-shot resolution of 100 ns can detect frequency changes of $1 \times 10^{-7}$ in 1 s. The current limit for TIC resolution is about 20 ps, which means that a frequency change of $2 \times 10^{-11}$ can be detected in 1 s. Averaging over longer intervals can improve the resolution to <1 ps in some units [6].

Since standard frequencies like 5 or 10 MHz are not practical to measure with a TIC, *frequency dividers* (shown in Figure 10.2) or *frequency mixers* are used to convert the test frequency to a lower frequency. Divider systems are simpler and more versatile, since they can be easily built or programmed to accommodate different frequencies. Mixer systems are more expensive, require more hardware including an additional reference oscillator, and can often measure only one input frequency (e.g., 10 MHz), but they have a higher signal-to-noise ratio than divider systems.

If dividers are used, measurements are made from the TIC, but instead of using these measurements directly, we determine the rate of change from reading to reading. This rate of change is called the *phase deviation*. We can estimate frequency offset as follows:

$$f(\text{offset}) = \frac{-\Delta t}{T}$$

where $\Delta t$ is the amount of phase deviation, and $T$ is the measurement period.

To illustrate, consider a measurement of +1 $\mu$s of phase deviation over a measurement period of 24 h. The unit used for measurement period (h) must be converted to the unit used for phase deviation ($\mu$s). The equation becomes

$$f(\text{offset}) = \frac{-\Delta t}{T} = \frac{-1\ \mu\text{s}}{86{,}400{,}000{,}000\ \mu\text{s}} = -1.16 \times 10^{-11}$$

As shown, a device that accumulates 1 $\mu$s of phase deviation/day has a frequency offset of $-1.16 \times 10^{-11}$ with respect to the reference. This simple example requires only two time interval readings to be made, and $\Delta t$ is simply the difference between the two readings. Often, multiple readings are taken and the frequency offset is estimated by using least squares linear regression on the data set, and obtaining $\Delta t$ from the slope of the least squares line. This information is usually presented as a phase plot, as shown in Figure 10.6. The device under test is high in frequency by exactly $1 \times 10^{-9}$, as indicated by a phase deviation of 1 ns/s [2,7,8].

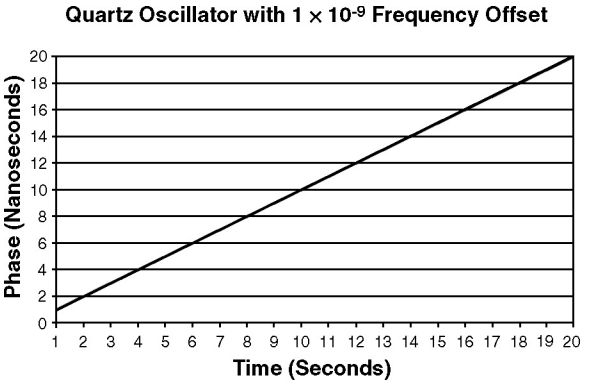



**FIGURE 10.6** A sample phase plot.

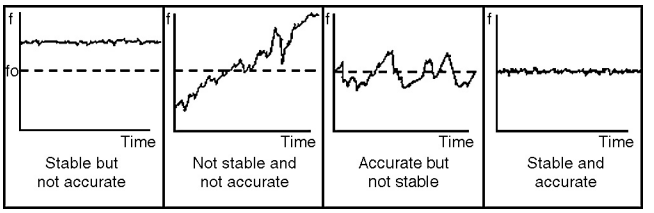



**FIGURE 10.7** The relationship between accuracy and stability.

Dimensionless frequency offset values can be converted to units of frequency (Hz) if the nominal frequency is known. To illustrate this, consider an oscillator with a nominal frequency of 5 MHz and a frequency offset of $+1.16 \times 10^{-11}$. To find the frequency offset in hertz, multiply the nominal frequency by the offset:

$$(5 \times 10^{6})\ (+1.16 \times 10^{-11}) = 5.80 \times 10^{-5} = +0.0000580 \text{ Hz}$$

Then, add the offset to the nominal frequency to get the actual frequency:

$$5{,}000{,}000 \text{ Hz} + 0.0000580 \text{ Hz} = 5{,}000{,}000.0000580 \text{ Hz}$$

## Stability

Stability indicates how well an oscillator can produce the same time or frequency offset over a given time interval. It doesn't indicate whether the time or frequency is "right" or "wrong," but only whether it *stays the same*. In contrast, accuracy indicates how well an oscillator has been set on time or on frequency. To understand this difference, consider that a stable oscillator that needs adjustment might produce a frequency with a large offset. Or, an unstable oscillator that was just adjusted might temporarily produce a frequency near its nominal value. Figure 10.7 shows the relationship between accuracy and stability.

Stability is defined as the statistical estimate of the frequency or time fluctuations of a signal over a given time interval. These fluctuations are measured with respect to a mean frequency or time offset. *Short-term* stability usually refers to fluctuations over intervals less than 100 s. *Long-term* stability can refer to measurement intervals greater than 100 s, but usually refers to periods longer than 1 day.

Stability estimates can be made in either the frequency domain or time domain, and can be calculated from a set of either frequency offset or time interval measurements. In some fields of measurement, stability is estimated by taking the standard deviation of the data set. However, standard deviation only works with stationary data, where the results are time independent, and the noise is *white*, meaning that

it is evenly distributed across the frequency band of the measurement. Oscillator data is usually nonstationary, since it contains time dependent noise contributed by the frequency offset. With stationary data, the mean and standard deviation will converge to particular values as more measurements are made. With nonstationary data, the mean and standard deviation never converge to any particular values. Instead, there is a moving mean that changes each time we add a measurement.

For these reasons, a non-classical statistic is often used to estimate stability in the time domain. This statistic is sometimes called the *Allan variance*, but since it is the square root of the variance, its proper name is the *Allan deviation*. The equation for the Allan deviation ($\sigma_y(\tau)$) is

$$\sigma_y(\tau) = \sqrt{\frac{1}{2(M-1)}\sum_{i=1}^{M-1}(y_{i+1}-y_i)^2}$$

where $y_i$ is a set of frequency offset measurements containing $y_1$, $y_2$, $y_3$, and so on, $M$ is the number of values in the $y_i$ series, and the data are equally spaced in segments $\tau$ seconds long. Or

$$\sigma_y(\tau) = \sqrt{\frac{1}{2(N-2)\tau^2}\sum_{i=1}^{N-2}[x_{i+2}-2x_{i+1}+x_i]^2}$$

where $x_i$ is a set of phase measurements in time units containing $x_1$, $x_2$, $x_3$, and so on, $N$ is the number of values in the $x_i$ series, and the data are equally spaced in segments $\tau$ seconds long. Note that while standard deviation subtracts the mean from each measurement before squaring their summation, the Allan deviation subtracts the previous data point. This differencing of successive data points removes the time dependent noise contributed by the frequency offset.

An Allan deviation graph is shown in Figure 10.8. It shows the stability of the device improving as the averaging period ($\tau$) gets longer, since some noise types can be removed by averaging. At some point, however, more averaging no longer improves the results. This point is called the *noise floor*, or the point where the remaining noise consists of nonstationary processes such as flicker noise or random walk. The device measured in Figure 10.8 has a noise floor of $\sim 5 \times 10^{-11}$ at $\tau = 100$ s.

Practically speaking, a frequency stability graph also tells us how long we need to average to get rid of the noise contributed by the reference and the measurement system. The noise floor provides some indication of the amount of averaging required to obtain a TUR high enough to show us the true frequency

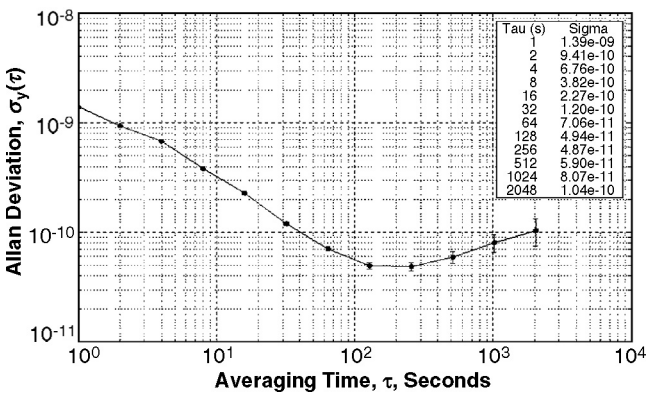



**FIGURE 10.8** A frequency stability graph.

**TABLE 10.2** Statistics Used to Estimate Time and Frequency Stability and Noise Types

| Name | Mathematical Notation | Description |
|---|---|---|
| Allan deviation | $\sigma_y(\tau)$ | Estimates frequency stability. Particularly suited for intermediate- to long-term measurements. |
| Modified Allan deviation | MOD $\sigma_y(\tau)$ | Estimates frequency stability. Unlike the normal Allan deviation, it can distinguish between white and flicker phase noise, which makes it more suitable for short-term stability estimates. |
| Time deviation | $\sigma_x(\tau)$ | Used to measure time stability. Clearly identifies both white and flicker phase noise, the noise types of most interest when measuring time or phase. |
| Total deviation | $\sigma_{y,\,\mathrm{TOTAL}}(\tau)$ | Estimates frequency stability. Particularly suited for long-term estimates where $\tau$ exceeds 10% of the total data sample. |

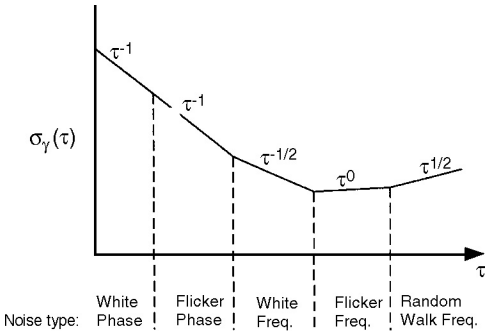



**FIGURE 10.9** Using a frequency stability graph to identify noise types.

offset of the DUT. If the DUT is an atomic oscillator (section 10.4) and the reference is a radio controlled transfer standard (section 10.5) we might have to average for 24 h or longer to have confidence in the measurement result.

Five noise types are commonly discussed in the time and frequency literature: *white phase, flicker phase, white frequency, flicker frequency*, and *random walk frequency*. The slope of the Allan deviation line can help identify the amount of averaging needed to remove these noise types (Figure 10.9). The first type of noise to be removed by averaging is phase noise, or the rapid, random fluctuations in the phase of the signal. Ideally, only the device under test would contribute phase noise to the measurement, but in practice, some phase noise from the measurement system and reference needs to be removed through averaging. Note that the Allan deviation does not distinguish between white phase noise and flicker phase noise. Table 10.2 shows several other statistics used to estimate stability and identify noise types for various applications.

Identifying and eliminating sources of oscillator noise can be a complex subject, but plotting the first order differences of a set of time domain measurements can provide a basic understanding of how noise is removed by averaging. Figure 10.10 was made using a segment of the data from the stability graph in Figure 10.8. It shows phase plots dominated by white phase noise (1 s averaging), white frequency noise (64 s averages), flicker frequency noise (256 s averages), and random walk frequency (1024 s averages). Note that the white phase noise plot has a 2 ns scale, and the other plots use a 100 ps scale [8–12].

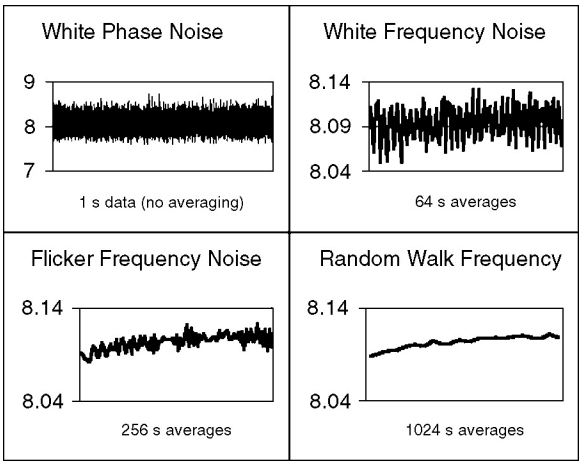



**FIGURE 10.10** Phase plots of four noise types.

## 10.3 Time and Frequency Standards

All time and frequency standards are based on a *periodic event* that repeats at a constant rate. The device that produces this event is called a *resonator*. In the simple case of a pendulum clock, the pendulum is the resonator. Of course, a resonator needs an energy source before it can move back and forth. Taken together, the energy source and resonator form an *oscillator*. The oscillator runs at a rate called the *resonance frequency*. For example, a clock's pendulum can be set to swing back and forth at a rate of once per second. Counting one complete swing of the pendulum produces a time interval of 1 s. Counting the total number of swings creates a *time scale* that establishes longer time intervals, such as minutes, hours, and days. The device that does the counting and displays or records the results is called a *clock*. Table 10.3 shows how the frequency uncertainty of a clock's resonator corresponds to the timing uncertainty of a clock.

Throughout history, clock designers have searched for more stable resonators, and the evolution of time and frequency standards is summarized in Table 10.4. The uncertainties listed for modern standards represent current (year 2001) devices, and not the original prototypes. Note that the performance of time and frequency standards has improved by 13 orders of magnitude in the past 700 years, and by about nine orders of magnitude in the past 100 years.

The stability of time and frequency standards is closely related to their quality factor, or $Q$. The $Q$ of an oscillator is its resonance frequency divided by its resonance width. The resonance frequency is the natural frequency of the oscillator. The resonance width is the range of possible frequencies where the oscillator will oscillate. A high-$Q$ resonator will not oscillate at all unless it is near its resonance frequency. Obviously, a high resonance frequency and a narrow resonance width are both advantages when seeking a high $Q$. Generally speaking, the higher the $Q$, the more stable the oscillator, since a high $Q$ means that an oscillator will stay close to its natural resonance frequency.

This section begins by discussing quartz oscillators, which achieve the highest $Q$ of any mechanical-type device. It then discusses oscillators with much higher $Q$ factors, based on the atomic resonance of rubidium and cesium. Atomic oscillators use the quantized energy levels in atoms and molecules as the source of their resonance. The laws of quantum mechanics dictate that the energies of a bound system, such as an atom, have certain discrete values. An electromagnetic field at a particular frequency can boost an atom from one energy level to a higher one. Or, an atom at a high energy level can drop to a lower level by emitting energy. The resonance frequency ($f$) of an atomic oscillator is the difference between

**Table 10.3** Relationship of Frequency Uncertainty to Time Uncertainty

| Frequency Uncertainty | Measurement Period | Time Uncertainty |
|---|---|---|
| $\pm 1.00 \times 10^{-3}$ | 1 s | ± 1 ms |
| $\pm 1.00 \times 10^{-6}$ | 1 s | ± 1 $\mu$s |
| $\pm 1.00 \times 10^{-9}$ | 1 s | ± 1 ns |
| $\pm 2.78 \times 10^{-7}$ | 1 h | ± 1 ms |
| $\pm 2.78 \times 10^{-10}$ | 1 h | ± 1 $\mu$s |
| $\pm 2.78 \times 10^{-13}$ | 1 h | ± 1 ns |
| $\pm 1.16 \times 10^{-8}$ | 1 day | ± 1 ms |
| $\pm 1.16 \times 10^{-11}$ | 1 day | ± 1 $\mu$s |
| $\pm 1.16 \times 10^{-14}$ | 1 day | ± 1 ns |

**TABLE 10.4** The Evolution of Time and Frequency Standards

| Standard | Resonator | Date of Origin | Timing Uncertainty (24 h) | Frequency Uncertainty (24 h) |
|---|---|---|---|---|
| Sundial | Apparent motion of the sun | 3500 B.C. | NA | NA |
| Verge escapement | Verge and foliet mechanism | 14th century | 15 min | $1 \times 10^{-2}$ |
| Pendulum | Pendulum | 1656 | 10 s | $1 \times 10^{-4}$ |
| Harrison chronometer (H4) | Spring and balance wheel | 1759 | 350 ms | $4 \times 10^{-6}$ |
| Shortt pendulum | Two pendulums, slave and master | 1921 | 10 ms | $1 \times 10^{-7}$ |
| Quartz crystal | Quartz crystal | 1927 | 10 $\mu$s | $1 \times 10^{-10}$ |
| Rubidium gas cell | $^{87}$Rb resonance (6,834,682,608 Hz) | 1958 | 100 ns | $1 \times 10^{-12}$ |
| Cesium beam | $^{133}$Cs resonance (9,192,631,770 Hz) | 1952 | 1 ns | $1 \times 10^{-14}$ |
| Hydrogen maser | Hydrogen resonance (1,420,405,752 Hz) | 1960 | 1 ns | $1 \times 10^{-14}$ |
| Cesium fountain | $^{133}$Cs resonance (9,192,631,770 Hz) | 1991 | 100 ps | $1 \times 10^{-15}$ |

the two energy levels divided by Planck's constant ($h$):

$$f = \frac{E_2 - E_1}{h}$$

The principle underlying the atomic oscillator is that since all atoms of a specific element are identical, they should produce exactly the same frequency when they absorb or release energy. In theory, the atom is a perfect "pendulum" whose oscillations are counted to measure time interval. The discussion of atomic oscillators is limited to devices that are commercially available, and excludes the primary and experimental standards found in laboratories such as NIST. Table 10.5 provides a summary [1,4,8].

## Quartz Oscillators

Quartz crystal oscillators are by far the most common time and frequency standards. An estimated two billion ($2 \times 10^9$) quartz oscillators are manufactured annually. Most are small devices built for wristwatches, clocks, and electronic circuits. However, they are also found inside test and measurement equipment, such as counters, signal generators, and oscilloscopes; and interestingly enough, inside every atomic oscillator.

**TABLE 10.5** Summary of Oscillator Types

| Oscillator Type | Quartz (TCXO) | Quartz (OCXO) | Rubidium | Commercial Cesium Beam | Hydrogen Maser |
|---|---|---|---|---|---|
| Q | $10^4$ to $10^6$ | $3.2 \times 10^6$ (5 MHz) | $10^7$ | $10^8$ | $10^9$ |
| Resonance frequency | Various | Various | 6.834682608 GHz | 9.192631770 GHz | 1.420405752 GHz |
| Leading cause of failure | None | None | Rubidium lamp (life expectancy >15 years) | Cesium beam tube (life expectancy of 3 to 25 years) | Hydrogen depletion (life expectancy >7 years) |
| Stability, $\sigma_y(\tau)$, $\tau = 1$ s | $1 \times 10^{-8}$ to $1 \times 10^{-9}$ | $1 \times 10^{-12}$ | $5 \times 10^{-11}$ to $5 \times 10^{-12}$ | $5 \times 10^{-11}$ to $5 \times 10^{-12}$ | $1 \times 10^{-12}$ |
| Noise floor, $\sigma_y(\tau)$ | $1 \times 10^{-9}$ ($\tau = 1$ to $10^2$ s) | $1 \times 10^{-12}$ ($\tau = 1$ to $10^2$ s) | $1 \times 10^{-12}$ ($\tau = 10^3$ to $10^5$ s) | $1 \times 10^{-14}$ ($\tau = 10^5$ to $10^7$ s) | $1 \times 10^{-15}$ ($\tau = 10^3$ to $10^5$ s) |
| Aging/year | $5 \times 10^{-7}$ | $5 \times 10^{-9}$ | $1 \times 10^{-10}$ | None | $\sim 1 \times 10^{-13}$ |
| Frequency offset after warm-up | $1 \times 10^{-6}$ | $1 \times 10^{-8}$ to $1 \times 10^{-10}$ | $5 \times 10^{-10}$ to $5 \times 10^{-12}$ | $5 \times 10^{-12}$ to $1 \times 10^{-14}$ | $1 \times 10^{-12}$ to $1 \times 10^{-13}$ |
| Warm-Up period | <10 s to $1 \times 10^{-6}$ | <5 min to $1 \times 10^{-8}$ | <5 min to $5 \times 10^{-10}$ | 30 min to $5 \times 10^{-12}$ | 24 h to $1 \times 10^{-12}$ |

A quartz crystal inside the oscillator is the resonator. It can be made of either natural or synthetic quartz, but all modern devices use synthetic quartz. The crystal strains (expands or contracts) when a voltage is applied. When the voltage is reversed, the strain is reversed. This is known as the *piezoelectric effect*. Oscillation is sustained by taking a voltage signal from the resonator, amplifying it, and feeding it back to the resonator. The rate of expansion and contraction is the resonance frequency and is determined by the cut and size of the crystal. The output frequency of a quartz oscillator is either the fundamental resonance or a multiple of the resonance, called an *overtone frequency*. Most high stability units use either the third or fifth overtone to achieve a high *Q*. Overtones higher than fifth are rarely used because they make it harder to tune the device to the desired frequency. A typical *Q* for a quartz oscillator ranges from $10^4$ to $10^6$. The maximum *Q* for a high stability quartz oscillator can be estimated as $Q = 1.6 \times 10^7/f$, where *f* is the resonance frequency in megahertz.

Environmental changes due to temperature, humidity, pressure, and vibration can change the resonance frequency of a quartz crystal, but there are several designs that reduce these environmental effects. The *oven-controlled crystal oscillator* (*OCXO*) encloses the crystal in a temperature-controlled chamber called an oven. When an OCXO is turned on, it goes through a "warm-up" period while the temperatures of the crystal resonator and its oven stabilize. During this time, the performance of the oscillator continuously changes until it reaches its normal operating temperature. The temperature within the oven then remains constant, even when the outside temperature varies. An alternate solution to the temperature problem is the *temperature-compensated crystal oscillator* (*TCXO*). In a TCXO, the signal from a temperature sensor is used to generate a correction voltage that is applied to a voltage-variable reactance, or varactor. The varactor then produces a frequency change equal and opposite to the frequency change produced by temperature. This technique does not work as well as oven control, but is less expensive. Therefore, TCXOs are used when high stability over a wide temperature range is not required.

Quartz oscillators have excellent short-term stability. An OCXO might be stable ($\sigma_y(\tau)$, at $\tau = 1$ s) to $1 \times 10^{-12}$. The limitations in short-term stability are due mainly to noise from electronic components in the oscillator circuits. Long-term stability is limited by *aging*, or a change in frequency with time due to internal changes in the oscillator. Aging is usually a nearly linear change in the resonance frequency that can be either positive or negative, and occasionally, a reversal in direction of aging occurs. Aging has many possible causes including a build-up of foreign material on the crystal, changes in the oscillator circuitry,

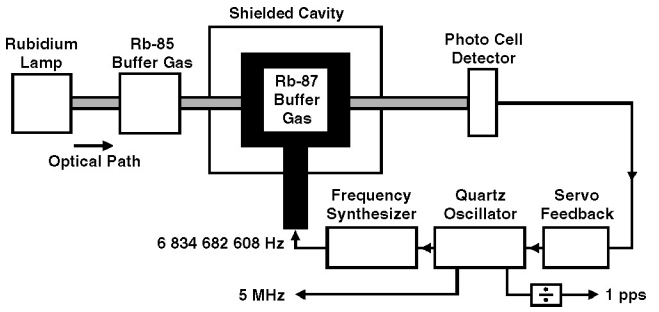



**FIGURE 10.11** Rubidium oscillator.

or changes in the quartz material or crystal structure. A high quality OCXO might age at a rate of $<5 \times 10^{-9}$ per year, while a TCXO might age 100 times faster.

Due to aging and environmental factors such as temperature and vibration, it is hard to keep even the best quartz oscillators within $1 \times 10^{-10}$ of their nominal frequency without constant adjustment. For this reason, atomic oscillators are used for applications that require better long-term accuracy and stability [4,13,14].

## Rubidium Oscillators

Rubidium oscillators are the lowest priced members of the atomic oscillator family. They operate at 6,834,682,608 Hz, the resonance frequency of the rubidium atom ($^{87}$Rb), and use the rubidium frequency to control the frequency of a quartz oscillator. A microwave signal derived from the crystal oscillator is applied to the $^{87}$Rb vapor within a cell, forcing the atoms into a particular energy state. An optical beam is then pumped into the cell and is absorbed by the atoms as it forces them into a separate energy state. A photo cell detector measures how much of the beam is absorbed, and its output is used to tune a quartz oscillator to a frequency that maximizes the amount of light absorption. The quartz oscillator is then locked to the resonance frequency of rubidium, and standard frequencies are derived from the quartz oscillator and provided as outputs (Figure 10.11).

Rubidium oscillators continue to get smaller and less expensive, and offer perhaps the best price-to-performance ratio of any oscillator. Their long-term stability is much better than that of a quartz oscillator and they are also smaller, more reliable, and less expensive than cesium oscillators.

The $Q$ of a rubidium oscillator is about $10^7$. The shifts in the resonance frequency are due mainly to collisions of the rubidium atoms with other gas molecules. These shifts limit the long-term stability. Stability ($\sigma_y(\tau)$, at $\tau = 1$ s) is typically $1 \times 10^{-11}$, and about $1 \times 10^{-12}$ at 1 day. The frequency offset of a rubidium oscillator ranges from $5 \times 10^{-10}$ to $5 \times 10^{-12}$ after a warm-up period of a few minutes or hours, so they meet the accuracy requirements of most applications without adjustment.

## Cesium Oscillators

*Cesium oscillators* are *primary frequency standards* since the SI second is defined from the resonance frequency of the cesium atom ($^{133}$Cs), which is 9,192,631,770 Hz. A properly working cesium oscillator should be close to its nominal frequency without adjustment, and there should be no change in frequency due to aging.

Commercially available oscillators use *cesium beam* technology. Inside a cesium oscillator, $^{133}$Cs atoms are heated to a gas in an oven. Atoms from the gas leave the oven in a high-velocity beam that travels through a vacuum tube toward a pair of magnets. The magnets serve as a gate that allows only atoms of a particular magnetic energy state to pass into a microwave cavity, where they are exposed to a microwave

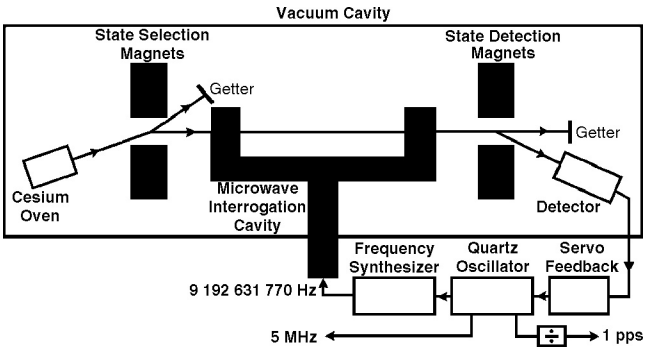



**FIGURE 10.12** Cesium beam oscillator.

frequency derived from a quartz oscillator. If the microwave frequency matches the resonance frequency of cesium, the cesium atoms change their magnetic energy state.

The atomic beam then passes through another magnetic gate near the end of the tube. Those atoms that changed their energy state while passing through the microwave cavity are allowed to proceed to a detector at the end of the tube. Atoms that did not change state are deflected away from the detector. The detector produces a feedback signal that continually tunes the quartz oscillator in a way that maximizes the number of state changes so that the greatest number of atoms reaches the detector. Standard output frequencies are derived from the locked quartz oscillator (Figure 10.12).

The $Q$ of a commercial cesium standard is a few parts in $10^8$. The beam tube is typically <0.5 m in length, and the atoms travel at velocities of >100 m/s inside the tube. This limits the observation time to a few milliseconds, and the resonance width to a few hundred hertz. Stability ($\sigma_y(\tau)$, at $\tau = 1$ s) is typically $5 \times 10^{-12}$ and reaches a noise floor near $1 \times 10^{-14}$ at about 1 day, extending out to weeks or months. The frequency offset is typically near $1 \times 10^{-12}$ after a warm-up period of 30 min.

## 10.4 Time and Frequency Transfer

Many applications require clocks or oscillators at different locations to be set to the same time (*synchronization*), or the same frequency (*syntonization*). *Time and frequency transfer* techniques are used to compare and adjust clocks and oscillators at different locations. Time and frequency transfer can be as simple as setting your wristwatch to an audio time signal, or as complex as controlling the frequency of oscillators in a network to parts in $10^{13}$.

Time and frequency transfer can use signals broadcast through many different media, including coaxial cables, optical fiber, radio signals (at numerous places in the radio spectrum), telephone lines, and the Internet. Synchronization requires both an on-time pulse and a time code. Syntonization requires extracting a stable frequency from the broadcast. The frequency can come from the carrier itself, or from a time code or other information modulated onto the carrier.

This section discusses both the fundamentals of time and frequency transfer and the radio signals used as calibration references. Table 10.6 provides a summary.

### Fundamentals of Time and Frequency Transfer

Signals used for time and frequency transfer are generally referenced to atomic oscillators that are steered to agree as closely as possible with UTC. Information is sent from a transmitter (A) to a receiver (B) and is delayed by $\tau_{ab}$, commonly called the *path delay* (Figure 10.13).

To illustrate path delay, consider a radio signal broadcast over a path 1000 km long. Since radio signals travel at the speed of light (~3.3 $\mu$s/km), we can calibrate the path by applying a 3.3-ms correction to our measurement. Of course, for many applications the path delay is simply ignored. For example, if our

**TABLE 10.6** Summary of Time and Frequency Transfer Signals and Methods

| Signal or Link | Receiving Equipment | Time Uncertainty (24 h) | Frequency Uncertainty (24 h) |
|---|---|---|---|
| Dial-Up Computer Time Service | Computer, client software, modem, and phone line | <15 ms | Not recommended for frequency measurements |
| Internet Time Service | Computer, client software, and Internet connection | <1 s | Not recommended for frequency measurements |
| HF Radio (3 to 30 MHz) | HF receiver and antenna | 1 to 20 ms | $10^{-6}$ to $10^{-9}$ |
| LF Radio (30 to 300 kHz) | LF receiver and antenna | 1 to 100 $\mu$s | $10^{-10}$ to $10^{-12}$ |
| Global Positioning System (GPS) | GPS receiver antenna | <20 ns | $<2 \times 10^{-13}$ |

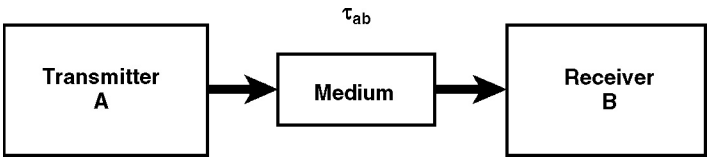



**FIGURE 10.13** One-way time and frequency transfer.

goal is simply to synchronize a computer clock within 1 s of UTC, there is no need to worry about a 100-ms path delay through a network. And, of course, path delay is not important to frequency transfer systems, since on-time pulses are not required. Instead, frequency transfer requires only a stable path where the delays remain relatively constant.

More sophisticated transfer systems estimate and remove all or part of the path delay. This is usually done in one of two ways. The first way is to estimate $\tau_{ab}$ and send the time out early by this amount. For example, if $\tau_{ab}$ is at least 20 ms for all users, the time can be sent 20 ms early. This advancement of the timing signal removes at least some of the delay for all users.

A better technique is to compute $\tau_{ab}$ and to apply a correction to the received signal. A correction for $\tau_{ab}$ can be computed if the position of both the transmitter and receiver are known. If the transmitter is stationary, a constant can be used for the transmitter position. If the transmitter is moving (a satellite, for example) it must broadcast its position in addition to broadcasting time. The Global Positioning System (GPS) provides the best of both worlds—each GPS satellite broadcasts its position and the receiver can use coordinates from multiple satellites to compute its own position.

The transmitted information often includes a *time code* so that a clock can be set to the correct time-of-day. Most time codes contain the UTC hour, minute, and second; the month, day, and year; and advance warning of daylight saving time and leap seconds.

## Radio Time and Frequency Transfer Signals

There are many types of radio receivers designed to receive time and frequency signals. Some are designed primarily to produce time-of-day information or an on-time pulse, others are designed to output standard frequencies, and some can be used for both time and frequency transfer. The following sections look at three types of time and frequency radio signals that distribute UTC—high frequency (HF), low frequency (LF), and GPS satellite signals.

### HF Radio Signals (Including WWV and WWVH)

High frequency (HF) radio broadcasts occupy the radio spectrum from 3 to 30 MHz. These signals are commonly used for time and frequency transfer at moderate performance levels. Some HF broadcasts provide audio time announcements and digital time codes. Other broadcasts simply provide a carrier frequency for use as a reference.

HF time and frequency stations include NIST radio stations WWV and WWVH. WWV is located near Fort Collins, Colorado, and WWVH is on the island of Kauai, Hawaii. Both stations broadcast continuous time and frequency signals on 2.5, 5, 10, and 15 MHz, and WWV also broadcasts on 20 MHz. All frequencies broadcast the same program, and at least one frequency should be usable at all times. The stations can also be heard by telephone; dial (303) 499-7111 for WWV or (808) 335-4363 for WWVH.

WWV and WWVH signals can be used in one of three modes:

- The audio portion of the broadcast includes seconds pulses or ticks, standard audio frequencies, and voice announcements of the UTC hour and minute. WWV uses a male voice, and WWVH uses a female voice.
- A binary time code is sent on a 100 Hz subcarrier at a rate of 1 bit per second. The time code contains the hour, minute, second, year, day of year, leap second and Daylight Saving Time (DST) indicators, and a UT1 correction. This code can be read and displayed by radio clocks.
- The carrier frequency can be used as a reference for the calibration of oscillators. This is done most often with the 5 and 10 MHz carrier signals, since they match the output frequencies of standard oscillators.

The time broadcast by WWV and WWVH will be late when it arrives at the user's location. The time offset depends upon the receiver's distance from the transmitter, but should be <15 ms in the continental United States. A good estimate of the time offset requires knowledge of HF radio propagation. Most users receive a signal that has traveled up to the ionosphere and was then reflected back to earth. Since the height of the ionosphere changes throughout the day, the path delay also changes. Path delay variations limit the received frequency uncertainty to parts in $10^9$ when averaged for 1 day.

HF radio stations such as WWV and WWVH are useful for low level applications, such as the manual synchronization of analog and digital clocks, simple frequency calibrations, and calibrations of stop watches and timers. However, LF and GPS signals are better choices for more demanding applications [2,7,15].

### LF Radio Signals (Including WWVB)

Before the advent of satellites, low frequency (LF) signals were the method of choice for time and frequency transfer. While the use of LF signals has diminished in the laboratory, they still have two major advantages—they can often be received indoors without an external antenna and several stations broadcast a time code. This makes them ideal for many consumer electronic products that display time-of-day information.

Many time and frequency stations operate in the LF band from 30 to 300 kHz (Table 10.7). The performance of the received signal is influenced by the path length and signal strength. Path length is important because the signal is divided into ground wave and sky wave. The ground wave signal is more stable. Since it travels the shortest path between the transmitter and receiver, it arrives first and its path delay is much easier to estimate. The sky wave is reflected from the ionosphere and produces results similar to those obtained with HF reception. Short paths make it possible to continuously track the ground wave. Longer paths produce a mixture of sky wave and ground wave. And over very long paths, only sky wave reception is possible.

Signal strength is also important. If the signal is weak, the receiver might search for a new cycle of the carrier to track. Each time the receiver adjusts its tracking point by one cycle, it introduces a phase step equal to the period of the carrier. For example, a cycle slip on a 60 kHz carrier introduces a 16.67 $\mu$s phase step. However, a strong ground wave signal can produce very good results. An LF receiver that

**TABLE 10.7** LF Time and Frequency Broadcast Stations

| Call Sign | Country | Frequency (kHz) | Always On? |
|---|---|---|---|
| DCF77 | Germany | 77.5 | Yes |
| DGI | Germany | 177 | Yes |
| HBG | Switzerland | 75 | Yes |
| JG2AS | Japan | 40 | Yes |
| MSF | United Kingdom | 60 | Yes |
| RBU | Russia | 66.666 | No |
| RTZ | Russia | 50 | Yes |
| TDF | France | 162 | Yes |
| WWVB | United States | 60 | Yes |

continuously tracks the same cycle of a ground wave signal can transfer frequency with an uncertainty of about $1 \times 10^{-12}$ when averaged for 1 day.

NIST operates LF radio station WWVB from Fort Collins, Colorado at a transmission frequency of 60 kHz. The station broadcasts 24 h per day, with an effective radiated output power of 50 kW. The WWVB time code is synchronized with the 60 kHz carrier and contains the year, day of year, hour, minute, second, and flags that indicate the status of daylight saving time, leap years, and leap seconds. The time code is received and displayed by wristwatches, alarm clocks, wall clocks, and other consumer electronic products [2,7,15].

### Global Positioning System (GPS)

The GPS is a navigation system developed and operated by the U.S. Department of Defense (DoD) that is usable nearly anywhere on the earth. The system consists of a constellation of at least 24 satellites that orbit the earth at a height of 20,200 km in six fixed planes inclined 55° from the equator. The orbital period is 11 h 58 m, which means that each satellite will pass over the same place on earth twice per day. By processing signals received from the satellites, a GPS receiver can determine its position with an uncertainty of <10 m.

The satellites broadcast on two carrier frequencies, L1 at 1575.42 MHz and L2 at 1227.6 MHz. Each satellite broadcasts a spread spectrum waveform, called a *pseudo random noise* (*PRN*) code on L1 and L2, and each satellite is identified by the PRN code it transmits. There are two types of PRN codes. The first type is a *coarse acquisition* (*C/A*) code with a chipping rate of 1023 chips per millisecond. The second is a *precision* (*P*) code with a chipping rate of 10230 chips per millisecond. The C/A code is broadcast on L1, and the P code is broadcast on both L1 and L2. GPS reception is line-of-sight, which means that the receiving antenna must have a clear view of the sky [16].

Each satellite carries either rubidium or cesium oscillators, or a combination of both. These oscillators are steered from DoD ground stations and are referenced to the United States Naval Observatory time scale, UTC (USNO), which by agreement is always maintained within 100 ns of UTC (NIST). The oscillators provide the reference for both the carrier and the code broadcasts.

GPS signals now dominate the world of high performance time and frequency transfer, since they provide reliable reception and exceptional results with minimal effort. A GPS receiver can automatically compute its latitude, longitude, and altitude from position data received from the satellites. The receiver can then calibrate the radio path and synchronize its on-time pulse. In addition to the on-time pulse, many receivers provide standard frequencies such as 5 or 10 MHz by steering an OCXO or rubidium oscillator using the satellite signals. GPS receivers also produce time-of-day and date information.

A GPS receiver calibrated for equipment delays has a timing uncertainty of <20 ns relative to UTC (NIST), and the frequency uncertainty is often $<2 \times 10^{-13}$ when averaged for 1 day. Figure 10.14 shows an Allan deviation plot of the output of a low cost GPS receiver. The stability is near $1 \times 10^{-13}$ after about 1 day of averaging.

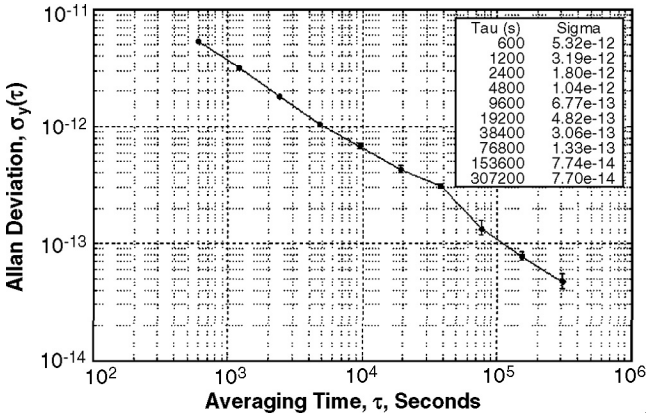



**FIGURE 10.14** Frequency stability of GPS receiver.

## 10.5 Closing

As noted earlier, time and frequency standards and measurements have improved by about nine orders of magnitude in the past 100 years. This rapid advance has made many new products and technologies possible. While it is impossible to predict what the future holds, we can be certain that oscillator Qs will continue to increase, measurement uncertainties will continue to decrease, and new technologies will continue to emerge.

### References

1. Jespersen, J., and Fitz-Randolph, J., *From Sundials to Atomic Clocks: Understanding Time and Frequency,* 2nd ed., Dover, Mineola, New York, 1999.
2. Kamas, G., and Lombardi, M. A., *Time and Frequency Users Manual,* NIST Special Publication 559, U.S. Government Printing Office, Washington, DC, 1990.
3. Levine, J., Introduction to time and frequency metrology, *Rev. Sci. Instrum.*, 70, 2567, 1999.
4. Hackman, C., and Sullivan, D. B., Eds., *Time and Frequency Measurement,* American Association of Physics Teachers, College Park, Maryland, 1996.
5. ITU Radiocommunication Study Group 7, *Selection and Use of Precise Frequency and Time Systems,* International Telecommunications Union, Geneva, Switzerland, 1997.
6. Novick, A. N., Lombardi, M. A., Zhang, V. S., and Carpentier, A., A high performance multi-channel time interval counter with an integrated GPS receiver, in *Proc. 31st Annu. Precise Time and Time Interval (PTTI) Meeting,* Dana Point, California, p. 561, 1999.
7. Lombardi, M. A., Time measurement and frequency measurement, in *The Measurement, Instrumentation, and Sensors Handbook,* Webster, J. G., Eds., CRC Press, Boca Raton, Florida, 1999, chap. 18–19.
8. Sullivan, D. B., Allan, D. W., Howe, D. A., and Walls, F. L., Eds., *Characterization of Clocks and Oscillators,* NIST Technical Note 1337, U.S. Government Printing Office, Washington, DC, 1990.
9. Jespersen, J., Introduction to the time domain characterization of frequency standards, in *Proc. 23rd Annu. Precise Time and Time Interval (PTTI) Meeting,* Pasadena, California, p. 83, 1991.
10. IEEE Standards Coordinating Committee 27, *IEEE Standard Definitions of Physical Quantities for Fundamental Frequency and Time Metrology—Random Instabilities,* Institute of Electrical and Electronics Engineers, New York, 1999.
11. Walls, F. L., and Ferre-Pikal, E. S., Measurement of frequency, phase noise, and amplitude noise, in *Wiley Encyclopedia of Electrical and Electronics Engineering,* John Wiley and Sons, New York, 1999, 12, 459.

12. Howe, D. A., An extension of the Allan variance with increased confidence at long term, *IEEE Int. Freq. Control Symp.,* 321, 1995.
13. Vig, J. R., Introduction to quartz frequency standards, *Army Research and Development Technical Report,* SLCET-TR-92-1, October 1992.
14. Hewlett-Packard Company, *Fundamentals of Quartz Oscillators,* HP Application Note 200-2, 1997.
15. Carr, J. J., *Elements of Electronic Instrumentation and Measurement,* 3rd ed., Prentice-Hall, NJ, 1996.
16. Hoffmann-Wellenhof, B., Lichtenegger, H., and Collins, J., *GPS: Theory and Practice, 3rd* ed., Springer-Verlag, New York, 1994.

# 11

# Sensor and Actuator Characteristics




Joey Parker
*University of Alabama*


Mechatronic systems use a variety of sensors and actuators to measure and manipulate mechanical, electrical, and thermal systems. Sensors have many characteristics that affect their measurement capabilities and their suitability for each application. Analog sensors have an output that is continuous over a finite region of inputs. Examples of analog sensors include potentiometers, LVDTs (linear variable differential transformers), load cells, and thermistors. Digital sensors have a fixed or countable number of different output values. A common digital sensor often found in mechatronic systems is the incremental encoder. An analog sensor output conditioned by an analog-to-digital converter (ADC) has the same digital output characteristics, as seen in Figure 11.1.

## 11.1 Range

The range (or span) of a sensor is the difference between the minimum (or most negative) and maximum inputs that will give a valid output. Range is typically specified by the manufacturer of the sensor. For example, a common type K thermocouple has a range of 800°C (from −50°C to 750°C). A ten-turn potentiometer would have a range of 3600 degrees.

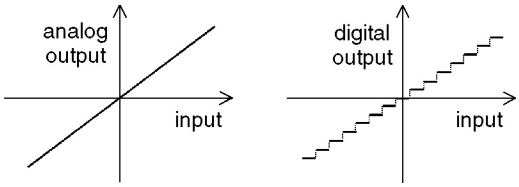



**FIGURE 11.1** Analog and digital sensor outputs.

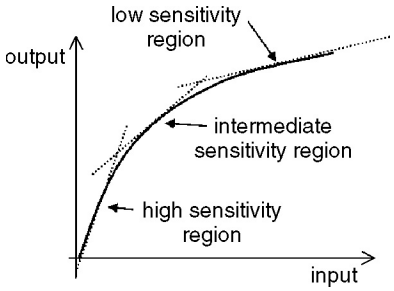



**FIGURE 11.2** Sensor sensitivity.

## 11.2 Resolution

The resolution of a sensor is the smallest increment of input that can be reliably detected. Resolution is also frequently known as the least count of the sensor. Resolution of digital sensors is easily determined. A 1024 ppr (pulse per revolution) incremental encoder would have a resolution of

$$\frac{1 \text{ revolution}}{1024 \text{ pulses}} \times \frac{360 \text{ degrees}}{1 \text{ revolution}} = 0.3516 \frac{\text{degrees}}{\text{pulse}}$$

The resolution of analog sensors is usually limited only by low-level electrical noise and is often much better than equivalent digital sensors.

## 11.3 Sensitivity

Sensor sensitivity is defined as the change in output per change in input. The sensitivity of digital sensors is closely related to the resolution. The sensitivity of an analog sensor is the slope of the output versus input line. A sensor exhibiting truly linear behavior has a constant sensitivity over the entire input range. Other sensors exhibit nonlinear behavior where the sensitivity either increases or decreases as the input is changed, as shown in Figure 11.2.

## 11.4 Error

Error is the difference between a measured value and the true input value. Two classifications of errors are bias (or systematic) errors and precision (or random) errors. Bias errors are present in all measurements made with a given sensor, and cannot be detected or removed by statistical means. These bias errors can be further subdivided into

- calibration errors (a zero or null point error is a common type of bias error created by a nonzero output value when the input is zero),
- loading errors (adding the sensor to the measured system changes the system), and
- errors due to sensor sensitivity to variables other than the desired one (e.g., temperature effects on strain gages).

## 11.5 Repeatability

Repeatability (or reproducibility) refers to a sensor's ability to give identical outputs for the same input. Precision (or random) errors cause a lack of repeatability. Fortunately, precision errors can be accounted for by averaging several measurements or other operations such as low-pass filtering. Electrical noise and hysteresis (described later) both contribute to a loss of repeatability.

## 11.6 Linearity and Accuracy

The accuracy of a sensor is inversely proportional to error, i.e., a highly accurate sensor produces low errors. Many manufacturers specify accuracy in terms of the sensor's linearity. A least-squares straight-line fit between all output measurements and their corresponding inputs determines the nominal output of the sensor. Linearity (or accuracy) is specified as a percentage of full scale (maximum valid input), as shown in Figure 11.3, or as a percentage of the sensor reading, as shown in Figure 11.4. Figures 11.3 and 11.4 show both of these specifications for 10% linearity, which is much larger than most actual sensors.

Accuracy and precision are two terms that are frequently confused. Figure 11.5 shows four sets of histograms for ten measurements of angular velocity of an actuator turning at a constant 100 rad/s. The first set of data shows a high degree of precision (low standard deviation) and repeatability, but the

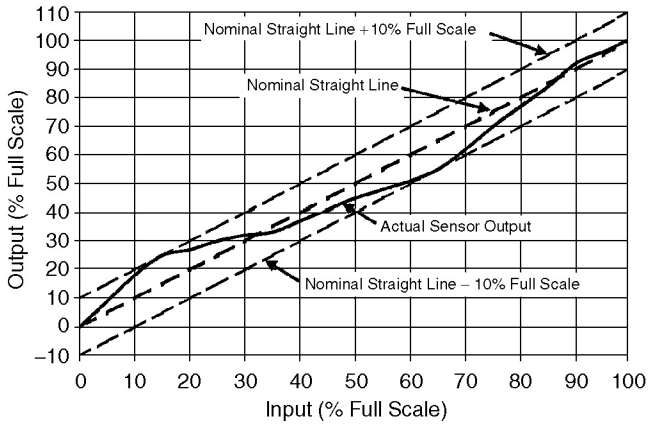



**FIGURE 11.3** Linearity specified at full scale.

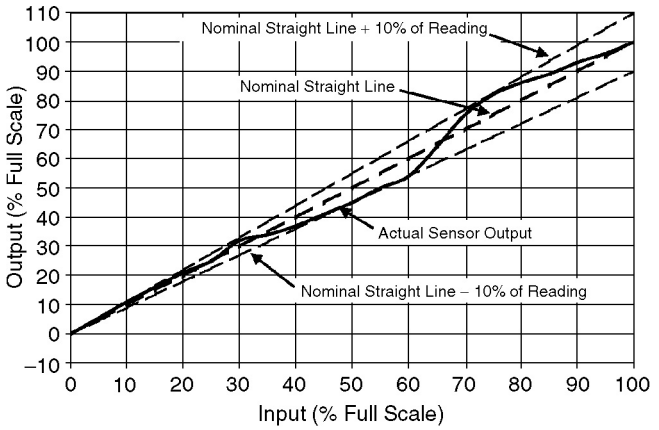



**FIGURE 11.4** Linearity specified at reading.

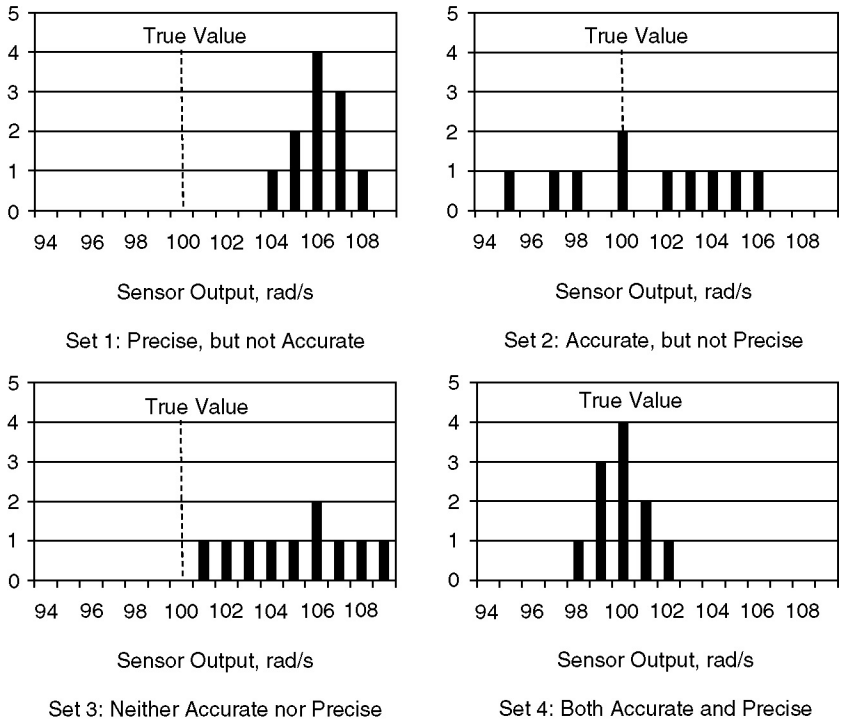



**FIGURE 11.5** Examples of accuracy and precision.

average accuracy is poor. The second set of data shows a low degree of precision (high standard deviation), but the average accuracy is good. The third set of data shows both low precision and low accuracy, while the fourth set of data shows both high precision, high repeatability, and high accuracy.

## 11.7 Impedance

Impedance is the ratio of voltage and current flow for a sensor. For a simple resistive sensor (such as a strain gage or a thermistor), the impedance $Z$ is the same as the resistance $R$, which has units of ohms ($\Omega$),

$$Z_R = \frac{V}{I} = R$$

For more complicated sensors, impedance includes the effects of capacitance, $C$, and inductance, $L$. Inclusion of these terms makes the impedance frequency sensitive, but the units remain ohms:

$$Z_C = \frac{V}{I} = \frac{1}{jC\omega} \qquad \text{and} \qquad Z_L = \frac{V}{I} = jL\omega$$

where $j = \sqrt{-1}$ is the imaginary number and $\omega$ is the driving frequency. The impedance form is particularly nice for analyzing simple circuits, as parallel and series inductances can be treated just like resistances. Two types of impedance are important in sensor applications: input impedance and output impedance. Input impedance is a measure of how much current must be drawn to power a sensor (or signal conditioning circuit). Input impedance is frequently modeled as a resistor in parallel with the

input terminals. High input impedance is desirable, since the device will then draw less current from the source. Oscilloscopes and data acquisition equipment frequently have input impedances of 1 MΩ or more to minimize this current draw. Output impedance is a measure of a sensor's (or signal conditioning circuit's) ability to provide current for the next stage of the system. Output impedance is frequently modeled as a resistor in series with the sensor output. Low output impedance is desirable, but is often not available directly from a sensor. Piezoelectric sensors in particular have high output impedances and cannot source much current (typically micro-amps or less). Op-amp circuits are frequently used to buffer sensor outputs for this reason. Op-amp circuits (especially voltage followers) provide nearly ideal circumstances for many sensors, since they have high input impedance but can substantially lower output impedance.

## 11.8 Nonlinearities

Linear systems have the property of superposition. If the response of the system to input A is output A, and the response to input B is output B, then the response to input C (= input A + input B) will be output C ( = output A + output B). Many real systems will exhibit linear or nearly linear behavior over some range of operation. Therefore, linear system analysis is correct, at least over these portions of a system's operating envelope. Unfortunately, most real systems have nonlinearities that cause them to operate outside of this linear region, and many common assumptions about system behavior, such as superposition, no longer apply. Several nonlinearities commonly found in mechatronic systems include static and coulomb friction, eccentricity, backlash (or hysteresis), saturation, and deadband.

## 11.9 Static and Coulomb Friction

In classic linear system analysis, friction forces are assumed to be proportional to velocity, i.e., viscous friction. With an actuator velocity of zero, there should be no friction. In reality, a small amount of static (no velocity) or Coulomb friction is almost always present, even in roller or ball type anti-friction bearings. A typical plot of friction force vs. velocity is given in Figure 11.6. Note that the static friction force can assume any value between some upper and lower limit at zero velocity. Static friction has two primary effects on mechatronic systems:

1. Some of the actuator torque or force is wasted overcoming friction forces, which leads to inefficiency from an energy viewpoint.
2. As the actuator moves the system to its final location, the velocity approaches zero and the actuator force/torque will approach a value that exactly balances frictional and gravity loads. Since static friction can assume any value at zero velocity, the actuator will come to slightly different final resting positions each time—depending on the final value of static friction. This effect contributes to some loss of repeatability in mechatronic systems.

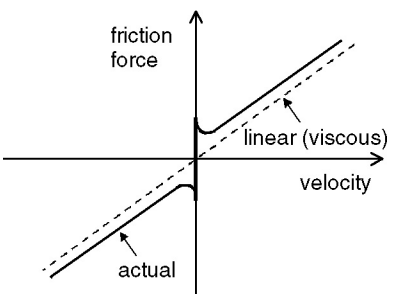



**FIGURE 11.6** Static and Coulomb friction.

## 11.10 Eccentricity

The ideal relationships for gears, pulleys, and chain drives assume that the point of gear contact remains at a fixed distance from the center of rotation for each gear. In reality, the true center of the gears pitch circle and the center of rotation will be separated by a small amount, known as the eccentricity. Small tooth-to-tooth errors can also cause local variations in the pitch circle radius. The combination of these two effects can lead to a nonlinear geometrical relationship between two gears like that of Figure 11.7, where the nonlinear behavior is greatly exaggerated for clarity. Eccentricity impacts the accuracy of position measurements made on the input side of the gear pair, as the output gear is not exactly where the sensor measurement indicates.

## 11.11 Backlash

If two otherwise perfect gears are not mounted on a center-to-center distance that exactly matches the sum of the pitch radii, there will be a small clearance, or backlash, between the teeth. When the input gear reverses direction, a small rotation is required before this clearance is removed and the output gear begins to move. Gear backlash is just one of many phenomena that can be characterized as hysteresis, as shown in Figure 11.8. Clearance between shafts and bearings can cause hysteretic effects also. Backlash exhibits effects similar to those for eccentricity, i.e., a loss of repeatability, particularly when approaching a measured point from different directions. The gear backlash problem is so prevalent and potentially harmful that many manufacturers go to great lengths to minimize or reduce the effect:

- gears mounted closer together than the theoretically ideal spacing,
- split "anti-backlash" gears that are spring loaded to force teeth to maintain engagement at all times,
- external spring-loaded mounts for one of the gears to force engagement, or
- specially designed gears with anti-backlash features.

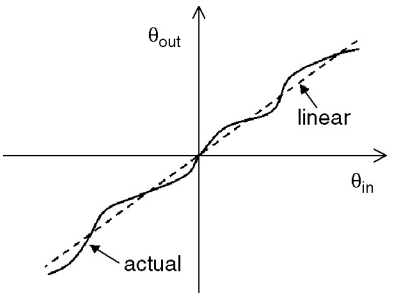



**FIGURE 11.7** Gear eccentricity.

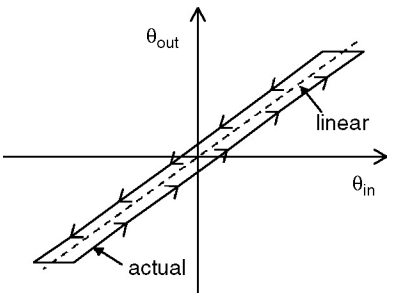



**FIGURE 11.8** Gear backlash.

## 11.12 Saturation

All real actuators have some maximum output capability, regardless of the input. This violates the linearity assumption, since at some point the input command can be increased without significantly changing the output; see Figure 11.9. This type of nonlinearity must be considered in mechatronic control system design, since maximum velocity and force or torque limitations affect system performance. Control systems modeled with linear system theory must be carefully tested or analyzed to determine the impact of saturation on system performance.

## 11.13 Deadband

Another nonlinear characteristic of some actuators and sensors is known as deadband. The deadband is typically a region of input close to zero at which the output remains zero. Once the input travels outside the deadband, then the output varies with input, as shown in Figure 11.10. Analog joystick inputs frequently use a small amount of deadband to reduce the effect of noise from human inputs. A very small movement of the joystick produces no output, but the joystick acts normally with larger inputs.

Deadband is also commonly found in household thermostats and other process type controllers, as shown in Figure 11.11. When a room warms and the temperature reaches the setpoint (or desired value)

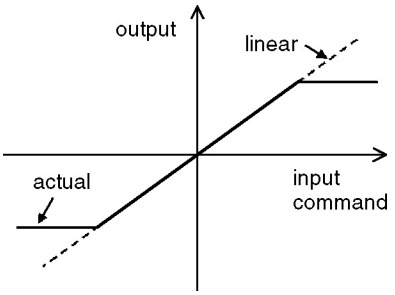



**FIGURE 11.9** Saturation.

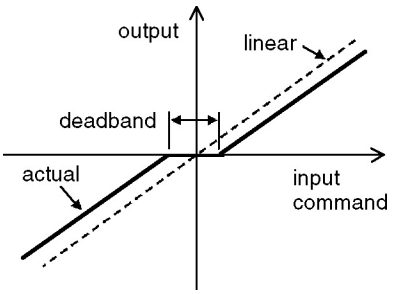



**FIGURE 11.10** Deadband.

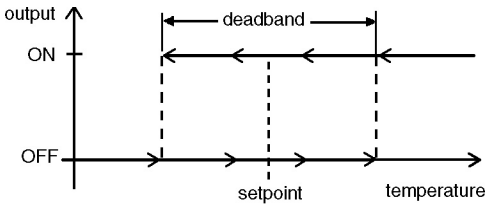



**FIGURE 11.11** Thermostat deadband.

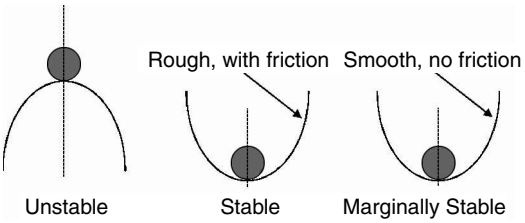



**FIGURE 11.12** System stability.

on the thermostat, the output remains off. Once room temperature has increased to the setpoint plus half the deadband, then the cooling system output goes to fully on. As the room cools, the output stays fully on until the temperature reaches the setpoint minus half the deadband. At this point the cooling system output goes fully off.

## 11.14 System Response

Sensors and actuators respond to inputs that change with time. Any system that changes with time is considered a dynamic system. Understanding the response of dynamic systems to different types of inputs is important in mechatronic system design. The most important concept in system response is stability. The term stability has many different definitions and uses, but the most common definition is related to equilibrium. A system in equilibrium will remain in the same state in the absence of external disturbances. A stable system will return to an equilibrium state if a "small" disturbance moves the system away from the initial state. An unstable system will not return to an equilibrium position, and frequently will move "far" from the initial state.

Figure 11.12 illustrates three stability conditions with a simple ball and hill system. In each case an equilibrium position is easily identified—either the top of the hill or the bottom of the valley. In the unstable case, a small motion of the ball away from the equilibrium position will cause the ball to move "far" away, as it rolls down the hill. In the stable case, a small movement of the ball away from the equilibrium position will eventually result in the ball returning, perhaps after a few oscillations. In the third case, the absence of friction causes the ball to oscillate continuously about the equilibrium position once a small movement has occurred. This special case is often known as marginal stability, since the system never quite returns to the equilibrium position.

Most sensors and actuators are inherently stable. However, the addition of active control systems can cause a system of stable devices to exhibit overall unstable behavior. Careful analysis and testing is required to ensure that a mechatronic system acts in a stable manner. The complex response of stable dynamic systems is frequently approximated by much simpler systems. Understanding both first-order and second-order system responses to either instantaneous (or step) changes in inputs or sinusoidal inputs will suffice for most situations.

## 11.15 First-Order System Response

First-order systems contain two primary elements: an energy storing element and an element which dissipates (or removes) energy. Typical first-order systems include resistor–capacitor filters and resistor–inductor networks (e.g., a coil of a stepper motor). Thermocouples and thermistors also form first-order systems, due to thermal capacitance and resistance. The differential equation describing the time response of a generic first-order system is

$$\frac{dy(t)}{dt} + \frac{1}{\tau} y(t) = f(t)$$

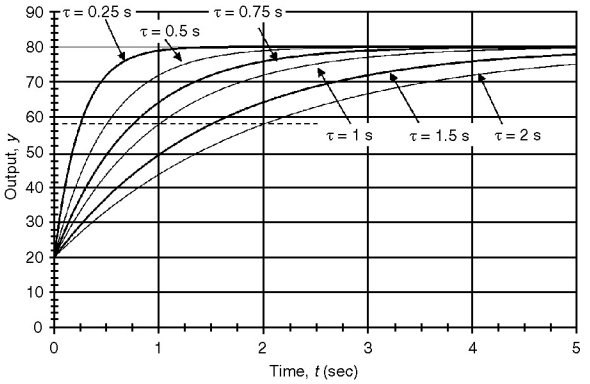



**FIGURE 11.13** First-order system—step response.

where $y(t)$ is the dependent output variable (velocity, acceleration, temperature, voltage, etc.), $t$ is the independent input variable (time), $\tau$ is the time constant (units of seconds), and $f(t)$ is the forcing function (or system input).

The solution to this equation for a step or constant input is given by

$$y(t) = y_\infty + (y_0 - y_\infty)e^{-t/\tau}$$

where $y_\infty$ is the limiting or final (steady-state) value, $y_0$ is the initial value of the independent variable at $t = 0$.

A set of typical first-order system step responses is shown in Figure 11.13. The initial value is arbitrarily selected as 20 with final values of 80. Time constants ranging from 0.25 to 2 s are shown. Each of these curves directly indicates its time constant at a key point on the curve. Substituting $t = \tau$ into the first-order response equation with $y_0 = 20$ and $y_\infty = 80$ gives

$$y(\tau) = 80 + (20 - 80)e^{-1} = 57.9$$

Each curve crosses the $y(\tau) \approx 57.9$ line when its time constant $\tau$ equals the time $t$. This concept is frequently used to experimentally determine time constants for first-order systems.

## 11.16 Underdamped Second-Order System Response

Second-order systems contain three primary elements: two energy storing elements and an element which dissipates (or removes) energy. The two energy storing elements must store different types of energy. A typical mechanical second-order system is the spring–mass–damper combination shown in Figure 11.14. The spring stores potential energy (PE = $\frac{1}{2}kx^2$), while the mass stores kinetic energy (KE = $\frac{1}{2}mv^2$), where $k$ is the spring stiffness (typical units of N/m), $x$ is the spring deflection (typical units of m), $m$ is the mass (typical units of kg), and $v$ is the absolute velocity of the mass (typical units of m/s).

A common electrical second-order system is the resistor–inductor–capacitor (RLC) network, where the capacitor and inductor store electrical energy in two different forms. The generic form of the dynamic equation for an underdamped second-order system is

$$\frac{d^2y(t)}{dt^2} + 2\zeta\omega_n\frac{dy(t)}{dt} + \omega_n^2 y(t) = f(t)$$

where $y(t)$ is the dependent variable (velocity, acceleration, temperature, voltage, etc.), $t$ is the independent variable (time), $\zeta$ is the damping ratio (a dimensionless quantity), $\omega_n$ is the natural frequency (typical units of rad/s), and $f(t)$ is the forcing function (or input).

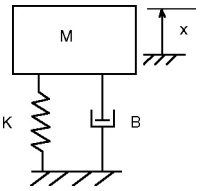



**FIGURE 11.14** Spring–mass–damper system.

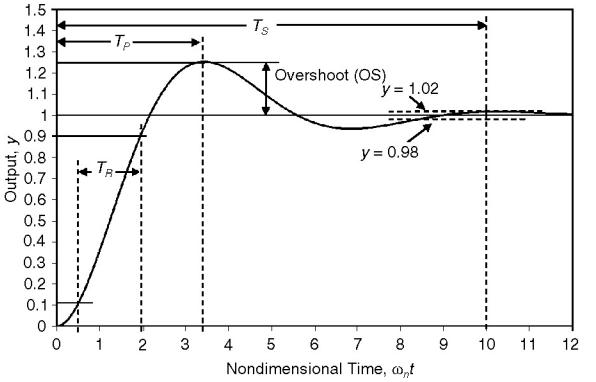



**FIGURE 11.15** Second-order system—step response.

The response of an underdamped ($0 \le \zeta < 1$) second-order system to a *unit* step input can be determined as:

$$y(t) = 1 - e^{-\zeta\omega_n t}\left(\cos\omega_n\sqrt{1-\zeta^2}\,t + \frac{\zeta}{\sqrt{1-\zeta^2}}\sin\omega_n\sqrt{1-\zeta^2}\,t\right)$$

This second-order system step response is often characterized by a set of time response parameters illustrated in Figure 11.15.

These time response parameters are functions of the damping ratio $\zeta$ and the natural frequency $\omega_n$:

- peak time, $T_P$: the time required to reach the first (or maximum) peak

$$T_P = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}}$$

- percent overshoot, %OS: amount the response exceeds or overshoots the steady-state value

$$\%\text{OS} = 100e^{-\left(\zeta\pi/\sqrt{1-\zeta^2}\right)}$$

- settling time, $T_S$: the time when the system response remains within ±2% of the steady-state value

$$T_S = \frac{4}{\zeta\omega_n}$$

- rise time, $T_R$: time required for the response to go from 10% to 90% of the steady-state value. Figure 11.16 shows the nondimensional rise time ($\omega_n T_R$) as a function of damping ratio, $z$. A frequently used approximation relating these two parameters is

$$\omega_n T_R \approx 2.16\zeta + 0.6 \quad 0.3 \le \zeta 0.8$$

Figures 11.17 and 11.18 show the unit step response of a second-order system as a function of damping ratio $\zeta$.

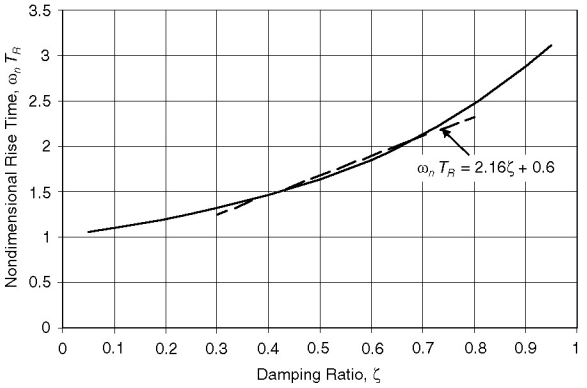



**FIGURE 11.16** Rise time vs. damping ratio, $\zeta$.

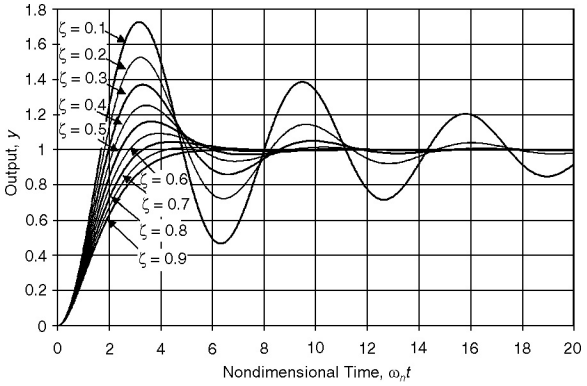



**FIGURE 11.17** Second-order system step response vs. damping ratio, $\zeta$.

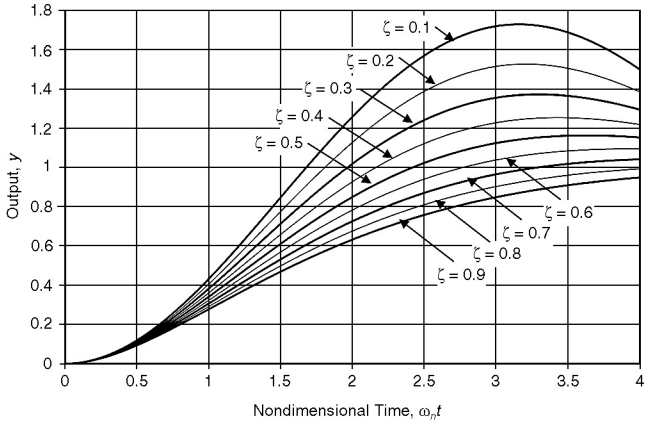



**FIGURE 11.18** Initial second-order system step response vs. damping ratio, $\zeta$.

## 11.17 Frequency Response

The response of any dynamic system to a sinusoidal input is called the frequency response. A generic first-order system with a sinusoidal input of amplitude A would have the dynamic equation of

$$\frac{dy(t)}{dt} + \frac{1}{\tau}\, y(t) \;=\; f(t) \;=\; A\sin(\omega t)$$

where $\omega$ is the frequency of the sinusoidal input and $\tau$ is the first-order time constant. The steady-state solution to this equation is

$$y(t) \;=\; AM\sin(\omega t + \Phi)$$

where $M \;=\; 1/\sqrt{(\tau\omega)^2 + 1}$ is the amplitude ratio (a dimensionless quantity), and $\Phi = -\tan^{-1}(\tau\omega)$ is the phase angle.

Figure 11.19 is a plot of the magnitude ratio $M_{\text{dB}}$ and the phase angle $\Phi$ as a function of the non-dimensional frequency, $\tau\omega$. Note that the magnitude is frequently plotted in terms of decibels, where $M_{\text{dB}} = 20\log_{10}(M)$.

The frequency at which the magnitude ratio equals 0.707 (or –3 dB) is called the bandwidth. For a first-order system, the bandwidth is inversely proportional to the time constant. So, $\omega = 1/\tau$.

A generic second-order system with a sinusoidal input of amplitude A and frequency $\omega$ would have the dynamic equation of

$$\frac{d^2y(t)}{dt^2} + 2\zeta\omega_n\frac{dy(t)}{dt} + \omega_n^2 y(t) \;=\; A\sin(\omega t)$$

The steady-state solution to this equation is

$$y(t) = \frac{AM}{\omega_n^2}\sin(\omega t + \Phi)$$

where

$$M \;=\; \frac{1}{\sqrt{[1-(\omega^2/\omega_n^2)]^2 + [2\zeta(\omega/\omega_n)]^2}}$$

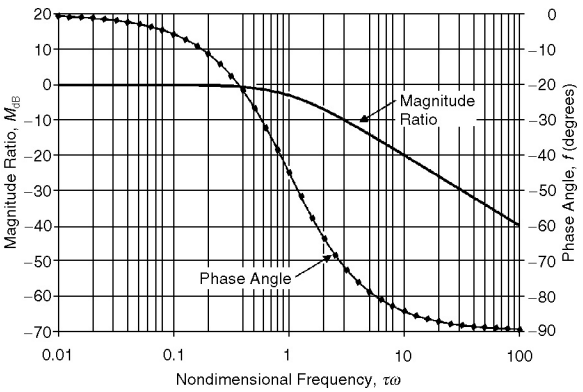



**FIGURE 11.19** Frequency response for first-order system.

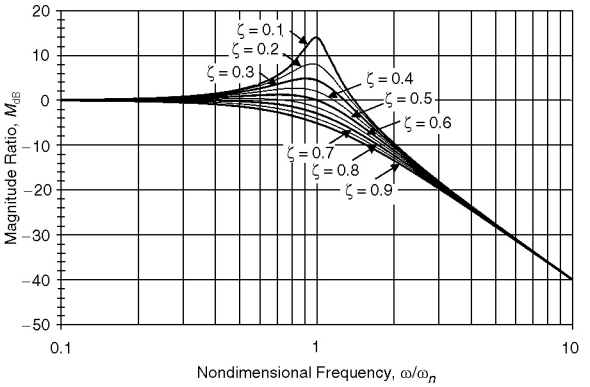



**FIGURE 11.20** Frequency response magnitude for second-order system.

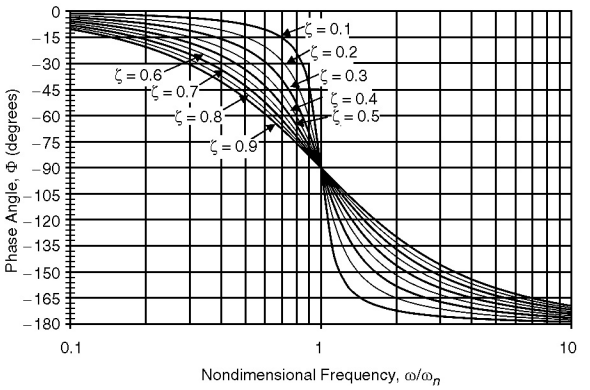



**FIGURE 11.21** Frequency response phase angle for second-order system.

is the amplitude ratio (a dimensionless quantity), and

$$\Phi = -\tan^{-1}\left[\frac{2\zeta(\omega/\omega_n)}{1-(\omega^2/\omega_n^2)}\right]$$

is the phase angle.

Figures 11.20 and 11.21 are plots of the magnitude response $M_{\text{dB}} = 20\log_{10}(M)$ and the phase angle $\Phi$ for the second-order system as a function of damping ratio, $\zeta$. The peak value in the magnitude response, $M_P$, can be found by taking the derivative of $M$ with respect to $\omega$ and setting the result to zero to find (Nise, 1995)

$$M_P = \frac{1}{2\zeta\sqrt{1-\zeta^2}}$$

This peak value in $M$ occurs at the frequency $\omega_P$ given by

$$\omega_P = \omega_n\sqrt{1-2\zeta^2}\,,$$

The peak value in an experimentally determined frequency response can be used to estimate both the natural frequency and damping ratio for a second-order system. These parameters can then be used to estimate time domain responses such as peak time and percent overshoot.

## Reference

Nise, N. S., *Control Systems Engineering,* 2nd ed., Benjamin/Cummings, 1995.

# 12

# The Role of Controls in Mechatronics




Job van Amerongen
*University of Twente*


## 12.1 Introduction

"Mechatronic design deals with the integrated and optimal design of a mechanical system and its embedded control system." This definition implies that the mechanical system is enhanced with electronic components in order to achieve a better performance, a more flexible system, or just reduce the cost of the system. In many cases the electronics are present in the form of a computer-based embedded (control) system. This does not imply that every controlled mechanical system is a mechatronic system because in many cases the control is just an add-on to the mechanical system in a sequential design procedure. A real mechatronics approach requires that an optimal choice be made with respect to the realization of the design specifications in the different domains. In control engineering the design of an optimal control system is well understood and for linear systems standard methods exist. The optimization problem is formulated as: given a process to be controlled, and given a performance index (cost function), find optimal controller parameters such that the cost function is minimized. With a state feedback controller and a quadratic cost function, solutions for the optimal controller gains can be found with standard controller design software, such as Matlab[1] (Figure 12.1).

Mechatronic design on the contrary requires that not only the controller be optimized. It requires optimization of the system as a whole. In the ideal case all the components in the system: the process itself, the controller, as well as the sensors and actuators, should be optimized simultaneously (Figure 12.2).

In general this is not feasible. The problem is ill defined and has to be split into smaller problems that can be optimized separately. Later on the partial solutions have to be combined and the performance of the complete system has to be evaluated. After eventually readjusting some parts of the system this leads to a sub-optimal solution.

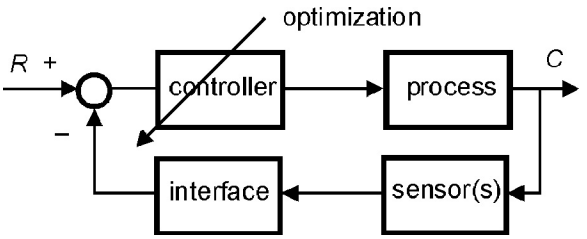



**FIGURE 12.1** Optimization of the controller.

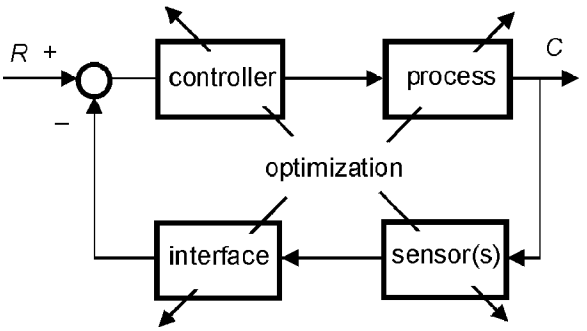



**FIGURE 12.2** Optimization of the all system components simultaneously.

In the initial conceptual design phase it has to be decided which problems should be solved mechanically and which problems electronically. In this stage decisions about the dominant mechanical properties have to be made, yielding a simple model that can be used for controller design. Also a rough idea about the necessary sensors, actuators, and interfaces has to be available in this stage. When the different partial designs are worked out in some detail, information about these designs can be used for evaluation of the complete system and be exchanged for a more realistic and detailed design of the different parts.

Although the word mechatronics is new, mechatronic products have been available for some time. In fact, all electronically controlled mechanical systems are based on the idea of improving the product by adding features realized in another domain. Good mechatronic designs are based on a *real systems approach*. But mostly, control engineers are confronted with a design in which major parameters are already fixed, often based on static or economic considerations. This prohibits optimization of the system as a whole, even when optimal control is applied.

In the last days of gramophones, the more sophisticated designs used tacho feedback in combination with a light turntable to achieve a constant number of revolutions. But a really new design was the compact disc player. Instead of keeping the number of revolutions of the disc constant, it aims for a constant speed of the head along the tracks of the disc. This means that the disc rotates slower when tracks with a greater diameter are read. The bits read from the CD are buffered electronically in a buffer that sends its information to the DA converter, controlled by a quartz crystal. This enables the realization of a very constant bit rate and eliminates all audible speed fluctuations. Such a performance could never be obtained from a pure mechanical device only, even if it were equipped with a good speed control system. In fact, the control loop for the disc speed does not need to have very strict specifications. It should only prevent overflow or underflow of the buffer. The high accuracy is obtained in an open loop mode, steered by a quartz crystal (Figure 12.3).

The flexibility introduced by the combination of precision mechanics and electronic control has allowed the development of CD-ROM players, running at speeds more than 50 times faster than the original audio CDs. A new way of thinking was necessary to come to such a new solution. On the other hand, the CD player is still a sophisticated piece of precision mechanics. No solid-state electronic memory

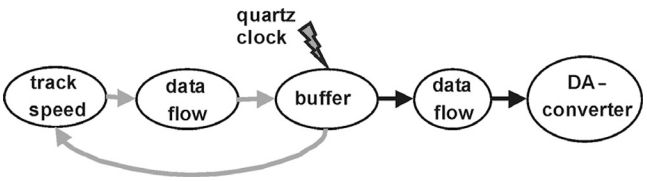



**FIGURE 12.3** Combination of closed-loop and open-loop control in a CD player.

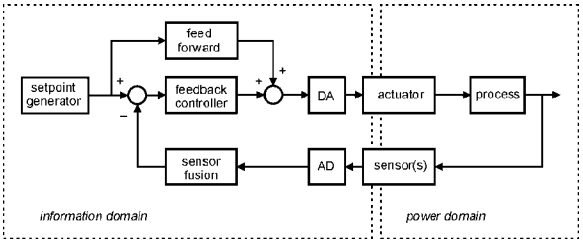



**FIGURE 12.4** Mechatronic system.

device can compete yet economically with the opto-mechanical storage capabilities of the CD and its successor the DVD. But this may change rapidly.

## 12.2 Key Elements of Controlled Mechatronic Systems

A mechatronic system consists by definition of a mechanical part that has to perform certain motions and an electronic part (in many cases an embedded computer system) that adds intelligence to the system. In the mechanical part of the system power plays a major role. This in contrast to the electronic part of the system where information processing is the main issue. Sensors convert the mechanical motions into electrical signals where only the information content is important or even into pure information in the form of numbers (if necessary, through an AD converter). Power amplifiers convert signals into modulated power. In most cases the power supply is electrical, but other sources such as hydraulic and pneumatic power supplies are possible as well. A controlled mechanical motion system thus typically consists of a mechanical construction, one or more actuators to generate the desired motions, and a controller that steers the actuators based on feed-forward and sensor-based feedback control (Figure 12.4).

## 12.3 Integrated Modeling, Design, and Control Implementation

### Modeling

During the design of mechatronic systems it is important that changes in the construction and the controller be evaluated simultaneously. Although a proper controller enables building a cheaper construction, a badly designed mechanical system will never be able to give a good performance by adding a sophisticated controller. Therefore, it is important that during an early stage of the design a proper choice can be made with respect to the mechanical properties needed to achieve a good performance of the controlled system. On the other hand, knowledge about the abilities of the controller to compensate for mechanic imperfections may enable that a cheaper mechanical construction be built. This requires that in an early stage of the design a simple model is available that reveals the performance limiting factors of the system. Still there is a gap between modeling and simulation software used for evaluation of mechanical constructions and software used for controller design. Mechanical engineers are used to

finite element packages to examine the dynamic properties of mechanical constructions. It is only after reduction to low-order models (modal analysis) that these models can be used for controller design. On the other hand, typical control-engineering software does not directly support the mechatronic design process either; in the modeling process the commonly used transfer functions and state space descriptions often have lost the relation with the physical parameters of the mechanical construction. Tools are required that allow modeling of mechanical systems in a way that the dominant physical parameters (like mass and dominant stiffness) are preserved in the model and simultaneously provide an interface to the controller design and simulation tools control engineers are used to (Coelingh;[2] Coelingh, De Vries, and Van Amerongen[3]).

Simulation is an important tool to evaluate the design of mechatronic systems. Most simulation programs like Simulink[1] use block diagram representations and do not support physical modeling in a way that direct tuning of the physical parameters of the mechanical construction and those of the controller is possible as required in the design of mechatronic systems. Recently, programs that allow physical modeling in *various physical domains* became available. They use an object-oriented approach that allows hierarchical modeling and reuse of models. The order of computation is only fixed after combining the subsystems. Examples of these programs are 20-sim,[4] described by Broenink[5] as CAMAS and Dymola.[6]

In this section the modeling and simulation program 20-sim (pronounced Twente Sim) will be used to illustrate the simultaneous design of construction and controller in a mechatronic system. 20-sim supports object-oriented modeling. Power and signal ports to and from the outside world determine each object (Weustink, De Vries, and Breedveld[7]). Inside the object there can be other objects or, on the lowest level, equations. Various *realizations* of an object can contain different or more detailed descriptions as long as the interface (number and type of ports) is identical. Modeling can start by a simple interconnection of (empty) submodels. Later they can be filled with realistic descriptions with various degrees of complexity. De Vries[8] refers to this as *polymorphic* modeling. Submodels can be constructed from other submodels in hierarchical structures. Proper physical modeling is achieved by coupling the submodels by means of the *flow of energy*, rather than by *signals* such as voltage, current, force, and speed. This way of modeling is well suited for mechatronics system design. It will be illustrated with an example. We want to consider the design of a simple servo system, considering the use of a voltage source, a DC motor, and a mechanical load driven through a transmission (Figure 12.5).

The transmission is disregarded for the time being. The belt is considered as infinitely stiff and the transformation ratio is taken care of by changing the motor constant. If a power amplifier driven by a signal generator describes the voltage source, we can draw the iconic diagram of Figure 12.6. At this stage the different *components* in this model are still empty. But all components have electrical and/or mechanical "ports." With the proper interfaces (ports) defined, the components can be connected to each other.

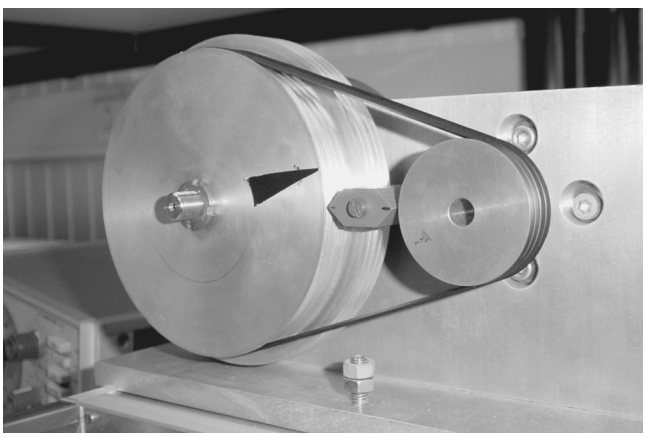


**FIGURE 12.5** Simple DC-servo system.

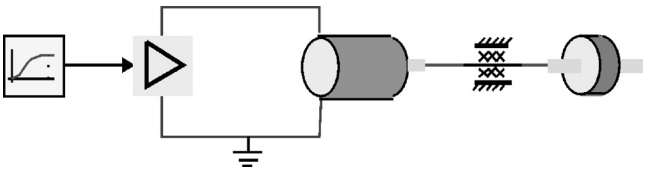


**FIGURE 12.6** Iconic diagram of the simple servo system.

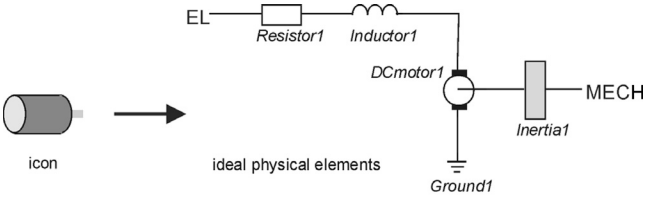



**FIGURE 12.7** Icon of the motor expanded to ideal physical elements.

In the next step we can detail the description of the DC motor. One solution could be the description given in Figure 12.7. The motor is now described by a number of *ideal physical elements*, each representing a basic physical relation. The motor has an electrical (EL) as well as a mechanical port (MECH).

Each of the *elements* in this figure can be described as an element with an electrical and/or mechanical port. The idea of ports is made more explicit in so-called bond graphs.[9–12] For the electrical elements these are the voltage difference over the element and the current through the element. For the mechanical elements these are the torque and the (angular) velocity. The products of these conjugated variables ($P = ui$ or $P = T\omega$) represent power.

If we go down a step further into the hierarchy, we arrive at the level of equations. For instance, an electrical resistor can be described by the equation:

$$p \cdot u = R * p \cdot i \tag{12.1}$$

where the variables $p \cdot u$ and $p \cdot i$ indicate the conjugated variables $u$ and $i$ of the electrical port $p$. Note that this is an equation and not an assignment statement. It could have been written equally well in the form:

$$p \cdot i = 1/R * p \cdot u \tag{12.2}$$

In a similar way the inductance can be described by the equations:

$$p \cdot u = L * \mathrm{ddt}(p \cdot i) \qquad \text{or} \qquad (p \cdot i) = 1/L * \mathrm{int}(p \cdot u) \tag{12.3}$$

where $\mathrm{ddt}(p \cdot i)$ denotes $di/dt$ and $\mathrm{int}(p \cdot u)$ denotes $\int u\,dt$. In case of an R-element there is no preference for one of the two forms. For the I-element the integral form is preferred in the simulations. 20-sim determines the preferred causal form and derives the equations automatically.

The energy flow or *power P* is the product of *two conjugated signals*, called effort ($e$) and flow ($f$):

$$P = ef \tag{12.4}$$

Examples of this expression in the mechanical and electrical domain are

$$P = Fv \qquad \text{or} \qquad P = T\omega \tag{12.5}$$

$$P = ui \tag{12.6}$$

where $F$ is force, $v$ is velocity, $T$ is torque, $\omega$ is angular velocity, $u$ is voltage, and $i$ is current.

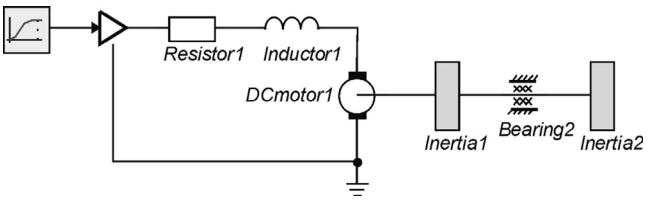



**FIGURE 12.8** Complete model in the form of ideal physical elements.

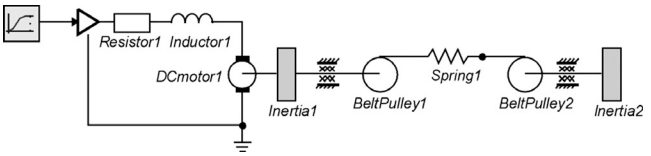



**FIGURE 12.9** Model extended with transmission.

When we expand the complete Figure 12.6 we obtain Figure 12.8. When this model is processed a message pops up that indicates that inertia 2 has a dependent state. The two inertias in this model always have the same speed, and therefore, they are dependent. They cannot have independent initial conditions. The message indicates that this element can only be written in derivative form:

$$T = J\, d\omega/dt \tag{12.7}$$

There are several ways to deal with this problem.

1. The two inertias can be combined into one inertia (the program will do this automatically). A message pops up that the dependency of the two inertias has been solved symbolically.
2. Dealing with the derivative causality by means of an implicit integration algorithm.
3. The transmission can be added, including some flexibility in the belt.

If the flexibility is negligible, solution 1 leads to the simplest model. On the other hand, the warning raises the question whether the flexibility of the belt can be disregarded indeed. If not, the model has to be extended with a spring element. It should be noted that this should not be done for numerical reasons only. If the transmission were very stiff, this would result in high-frequency dynamics and lead to unnecessary slow simulations. On the other hand, if the flexibility is important, as it is in this system, the warning draws the designer's attention to the fact that the model may be oversimplified. In Figure 12.9 the transmission, including a spring element, has been added. Processing of this model does not produce any warnings.

This example illustrates how modern software can help to come up with a model that has the complexity that is needed for a particular problem. Physical models, in the form of an iconic diagram, based on connecting elements by means of power ports, may help in this modeling process. The user can select the preferred view, whether this is a bond graph, an iconic diagram with ideal physical element, or a view using higher lever submodels, like in Figure 12.6. In the next section it will be shown how to use this model for the design of controllers.

## Control System Design Methodologies

Many processes can be reasonably well controlled by means of PID controllers. This is due to the fact that these processes can be more or less accurately described by means of a second-order model. Tuning rules, like those of Ziegler Nichols, enable less experienced people to tune such controllers. Relatively simple

models can also describe many mechatronic systems. A mechatronic system mostly consists of an actuator, some form of transmission, and a load. A fourth-order model can properly describe such a system. The performance-limiting factor in these systems is the resonance frequency. A combination of position and tacho feedback (basically a PD controller) can be applied here as well. But due to the resonant poles proper selection of the signals to be used in the feedback is essential. Efforts have been made (Groenhuis;[13] Coelingh;[2] Coelingh, De Vries, and Van Amerongen[3]) to derive recipes for tuning such systems, in addition to selecting the proper feedback signals. Computer support tools are essential to enable less experienced designers to use these recipes (Van Amerongen, Coelingh, and De Vries[14]). Coelingh[2] and Coelingh, De Vries, and Van Amerongen[3] describe a structural design method for mechatronic systems. The method starts with reducing the conceptual design to a fourth-order model that represents the dominant properties of the system in terms of the total mass to be moved and the dominant stiffness. This model still has physical meaningful parameters. In this model appropriate sensors are chosen, as well as a path generator. In the conceptual design phase a simple controller is developed and mechanical properties are changed, if necessary. Then a more detailed design phase follows where also parameter uncertainties are taken into account.

## Servo System Design

Here we will consider some simple aspects of the design of a servo system in order to illustrate the advantage of the use of physical models and to illustrate the need for an integrated design approach. We consider the model discussed before, a load driven by an electric motor, through a flexible transmission. The iconic diagram of this model was given in Figure 12.9. In this example a current amplifier has replaced the voltage amplifier allowing the removal of the electrical resistor and the inductance. In the step responses of Figure 12.10 the resonance due to the flexible transmission is clearly visible.

From the equations used for the simulation, 20-sim can automatically derive a model in a form suitable for controller design, such as a state-space description, a transfer function, or poles and zeros. An interface is provided to Matlab[1] enabling, for instance, to use Matlab algorithms to compute the gains of advanced controllers like an LQR (optimal state feedback) or LQG controller (with a Kalman filter for state estimation and optimal state feedback). The diagram of the process together with an LQG controller is given in Figure 12.11 and some responses in Figure 12.12.

A properly designed P(I)D controller is able to perform almost similarly, especially when the amount of noise is small. A first attempt could be to use only measurements of the load angle and load speed.

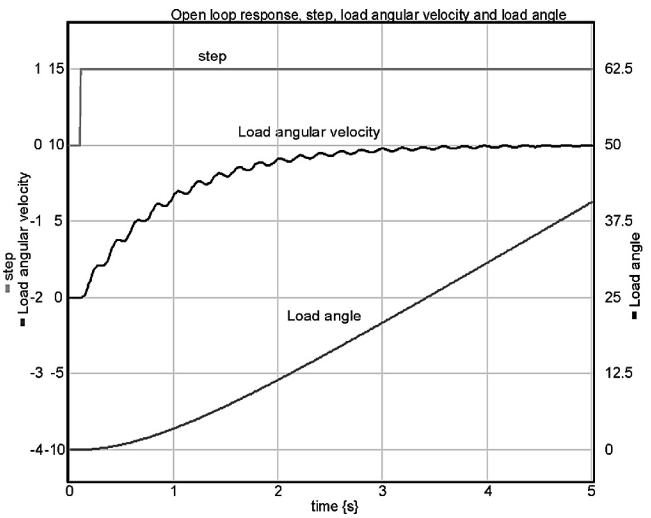



**FIGURE 12.10** Open loop responses.

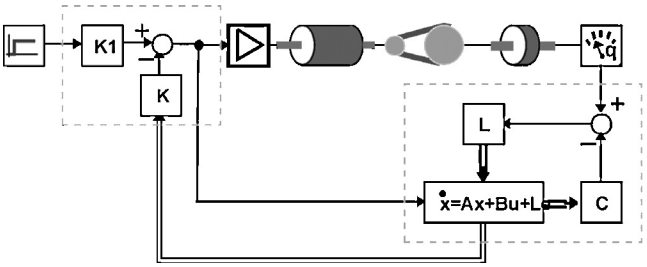



**FIGURE 12.11** Process with Kalman filter and state feedback.

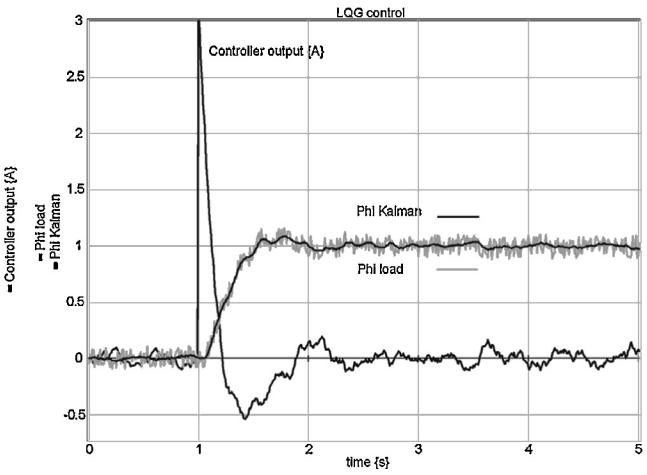



**FIGURE 12.12** Response of the LQG-controlled system.

This attempt fails, because feedback of the load speed leads almost immediately to an unstable system as can be seen from the root locus for variations in the gain of the velocity feedback. From the responses of Figure 12.10, 20-sim can easily determine the transfer function between the motor current and the load speed and plot the root locus (Figure 12.13).

Figure 12.14 clearly shows that even a small amount of velocity feedback will lead to an unstable system. It is well known that feedback of the motor speed is a better solution. Using again the model of Figures 12.9 and 12.10 to determine the transfer from input current to motor speed yields the root locus of Figure 12.15.

Complex zeros now accompany the complex poles and because they are close together their influence on the response will be almost negligible. The branch of the root locus on the real axis now shows the desired behavior: moving the dominant pole to the left in the *s*-plane. Combining the feedback of the motor speed with feedback of the load angle yields the PD-controller structure of Figure 12.15 and the responses of Figure 12.16. Except for the noise there is not much difference in the responses of the system with the Kalman filter, although the PD-controlled system is simpler. The observations made here are generally applicable. A system with two resonant (complex) poles and no zeros, such as in Figure 12.13, is difficult to control by means of a simple controller. If complex zeros accompany the resonant poles with an imaginary part smaller than that of the poles, stable control is easily achieved. In the frequency domain this is seen as an anti-resonance, followed by a resonance (type *AR*). On the contrary a type *RA* system, where the resonance frequency is lower than the anti-resonance frequency (the imaginary part of the poles is smaller than that of the zeros), is just as difficult to control as in the case of only resonant poles. The existence and location of resonant zeros is completely determined by the (geometrical) location of

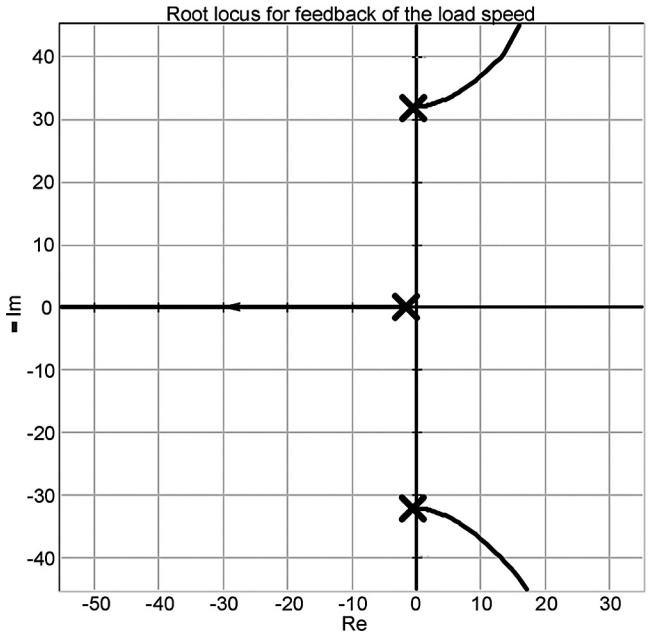



**FIGURE 12.13** Root locus for velocity feedback of load axis.

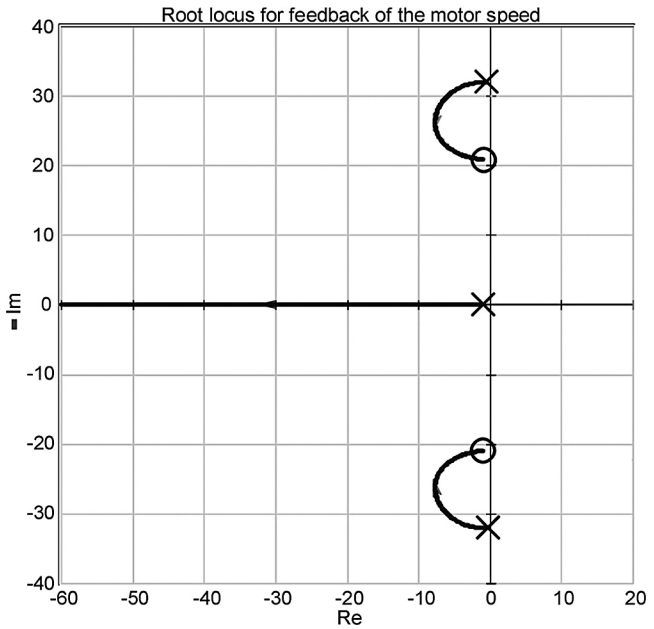



**FIGURE 12.14** Root locus for velocity feedback of motor axis.

the sensors in the mechanical system. A careful choice of these sensor locations is therefore crucial for the successful application of a controller. It should be noted that using a properly designed set-point generator could prevent resonance, as seen in Figure 12.10. The set point generator should not excite the resonance frequencies, for instance, by using a low pass filter with bandwidth lower than the resonance frequencies. However, such a set-point generator does not solve the above-mentioned stability problems.

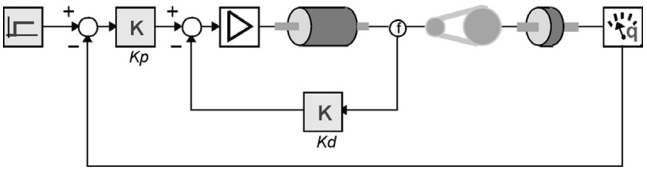



**FIGURE 12.15** Servo system with PD-controller.

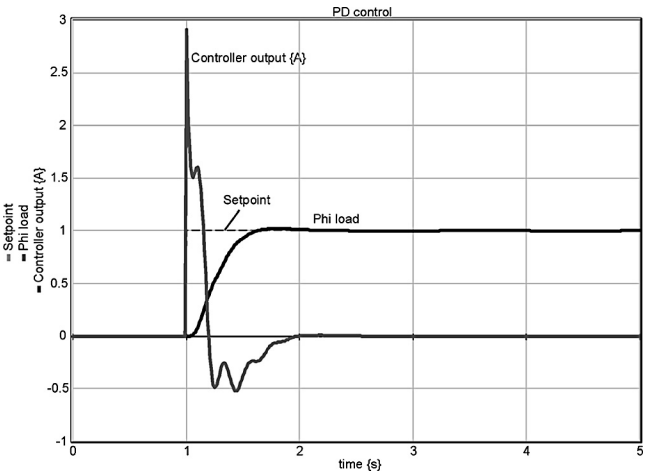



**FIGURE 12.16** Responses of the system of Figure 12.16.

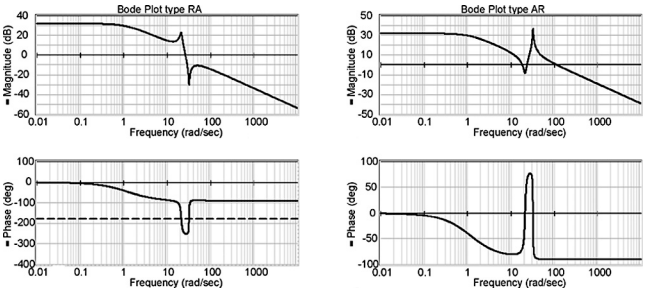



**FIGURE 12.17** Bode plots of type *RA* and *AR* systems.

## Design of a Mobile Robot

A typical example of the early design procedure is the conceptual design of a mobile assembly robot. Already in a very early stage of the design conflicting demands have to be resolved. Such a robot should be able to collect parts all around a production facility and do the assembly while driving. Because a high accuracy is required between the gripper of the robot and the surface where the parts are located, it is important that floor irregularities and vibration modes of the structure do not prevent proper assembly. On the other hand the path controller, partly based on dead reckoning (i.e., measuring of the wheel speed and orientation), requires that the wheels be very stiff. Damping of disturbances has to be realized by another means of suspension. This has led to the concept of an upper frame and a lower frame, connected by means of springs (Figure 12.18).

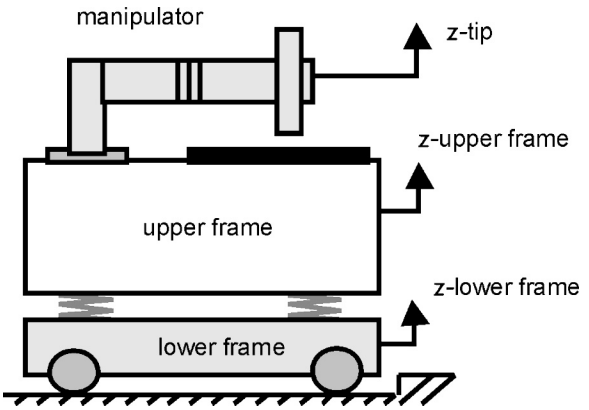



**FIGURE 12.18** Conceptual design of the mobile robot.

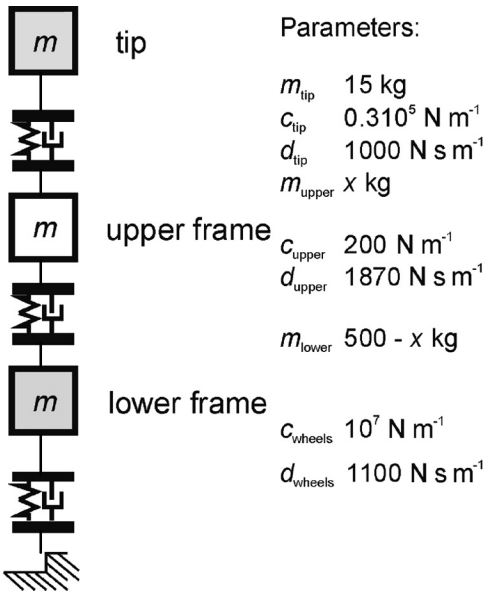



**FIGURE 12.19** Simple model with ideal physical elements to compute the error $e_{tip}$.

The robot can be mounted at the upper frame and should have sufficient bandwidth such that the position error ($e_{tip} = z_{tip} - z_{upper\ frame}$) between the tip of the robot ($z_{tip}$) and the upper frame ($z_{upper\ frame}$) is small enough.

The next step is to derive a simple model, in order to have some parameters for the weight distribution and the stiffness and damping of the springs. In the model of Figure 12.22 the robot is confronted with a bump in the floor at a speed of 1 m/s.

Based upon the payload—mainly the weight of the batteries—the total mass of the vehicle was estimated to be 500 kg. Stiffness and damping of the wheels follow from the demands for the accuracy of the position estimation. The mass and bandwidth of the controlled manipulator were already known from other studies, yielding the effective stiffness and damping for the robot tip. When also initial estimates of the stiffness and damping of the springs between the upper and lower frame are made, the only parameter to be varied is the weight distribution between the upper frame and lower frame. By using the optimization feature of 20-sim, the optimal weight distribution can easily be found. In order to minimize the error between the tip of the robot and the upper frame (Figure 12.19), the weight has to be placed as much as possible in

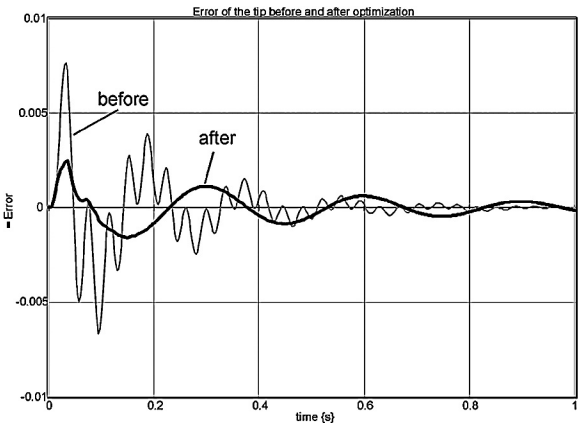



**FIGURE 12.20** Error of the tip before and after optimization of the weight distribution between upper and lower frame.

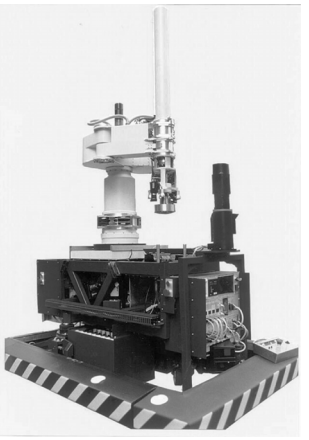


**FIGURE 12.21** The mobile robot (MART) after completion.

the upper frame (Figure 12.20). This example illustrates how the mechanical configuration of the system is determined by the requirements for good path control and accurate control of the assembly task.

A next step could be to optimize the properties of the suspension between upper and lower frame. This will further improve the error. This decision made in a very early stage of the design directed other design decisions. After completion of the project it appeared that the different parameters of the final construction were close to these early estimates (Figure 12.21).

## 12.4 Modern Examples of Mechatronic Systems in Action

A few examples have already been treated in the previous sections. In this section two more examples will be given.

### Rudder Roll Stabilization of Ships

Nowadays most ships use an autopilot to control the heading of the ship. A rudder is the most commonly used actuator. Some ships, like ferries and naval ships, need also roll stabilization. This can be achieved passively by means of two connected tanks filled with water that generate stabilizing forces that should

be in counter phase with the forces of the waves. In order to make the system effective for varying frequencies of the waves, the water flow between the two tanks should be controlled. For fast ships mostly stabilizing fins are used. These are a kind of actively controlled "wings" that generate the moments needed to counteract the moments of the waves. The fins not only influence the roll motions but also have influence on the heading. On the other hand, the rudder not only influences the heading but also induces roll. In control engineering terms this leads to a multivariable system that requires a multivariable controller design for optimum performance. In practice such a multivariable system is seldom seen and two separate control systems are used.

Another approach is to use only one of the actuators (rudder or fins) to achieve course control and roll reduction. Because the frequencies of the roll motions are outside the bandwidth of the course-control system this is possible. The rudder is most suited as actuator. An additional advantage for naval ships is that removing the fins will reduce the underwater noise of the vessel.

Redesigning the course controller in order to stabilize the roll as well, demonstrates the feasibility of this approach, but also makes clear that the "process"—the ship—should be modified. The most important modification is needed for the steering machine. The maximum speed of the steering machine appears to be the limiting factor for such a system (it should increase from the commonly used values of 3–7°/s to 20–25°/s). By means of dynamic simulations the demands for the steering machine can be found in terms of the maximum speed of the steering machine and the maximum time constant that is allowed for reaching this speed. This requires reengineering of the hydraulic steering machine. A step further would be to consider also changes in the shape of the ship, in order to optimize the parameters that determine the effectiveness of the rudder roll stabilization system.

In order to decide whether this new solution is better, it should be evaluated whether the redesigned steering machine is less expensive than the original rudder and fin actuators. These design issues have to be solved in a very early stage of the design. Rudder roll stabilization has been successfully applied on naval as well as merchant marine ships (Van Amerongen, Van der Klugt, and Van Nauta Lemke[15]).

## Compensation of Nonlinear Effects in a Linear Motor

Many mechanical systems suffer from nonlinear effects that limit the accuracy that can be achieved. Friction and cogging are two examples. A (linear) feedback controller can diminish the influence of non-linearities, but complete compensation may be difficult. For systems that perform repetitive motions, an Iterative Learning Controller can help to further improve the performance (Arimoto,[16] De Vries, Velthuis, and Van Amerongen[17]). The basic idea is explained in Figure 12.22.

When only the feedback loop is present and under the assumption that there are no disturbances, the error signal and thus the controller signal $U_C$ will be the same for each repetitive motion. It is obvious that the accuracy can be improved when in the next motion the controller signal from the former cycle is used as a feed-forward signal, $U_F$. The feedback will generate a signal that further compensates for the remaining error by updating the feed-forward signal $U_F$ with the formula

$$U_F^{k+1} = U_F^k + LE^k \tag{12.8}$$

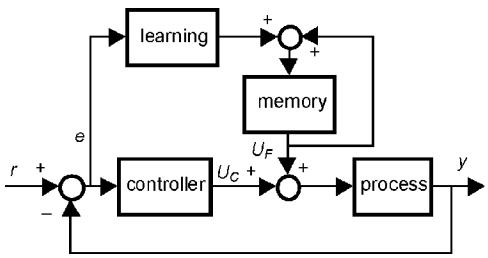



**FIGURE 12.22** Principle of Iterative Learning Control.

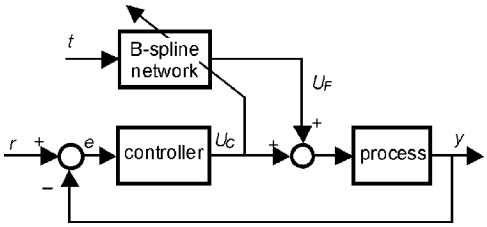



**FIGURE 12.23** Learning feed-forward controller for repetitive motions.

where $L$ is the transfer function of the learning filter. The superscript $k$ denotes the $k$th repetitive motion. The signal $U_F$ should converge to a feed-forward signal that compensates for all repetitive errors. An example of a situation where such errors are present is, for instance, a CD player that has to compensate for the eccentricity of the disk.

A variation on this idea and even more straightforward is the learning feed-forward controller (LFFC) setup of Figure 12.23. When the feed-forward signal would be perfect, the output of the controller would be zero. This implies that this output can be used as a training signal for a neural network. An adaptive B-spline network enables learning of complex nonlinear characteristics. Also support vector machines have been used to implement the learning feed forward (De Kruif and De Vries[18]). The input of the B-spline network is the time $t$. It is reset each time a new motion starts. This is called a time-indexed LFFC. Instead of the time, also the reference signal and its derivatives—obtained from a path generator—could be used as index for the network (path-indexed LFFC). The advantage of this structure is that after proper training the LFFC can successfully be used for nonrepetitive motions as well. Velthuis[19] has given a stability analysis for time-indexed as well as path-indexed LFFC-controllers. The stability analysis is relatively easy for the time-indexed case. For the path-indexed case it is more complex and some heuristics are required to guarantee a stable system. The main issue is that the number of B-splines should not be too large. On the other hand a sufficiently dense B-spline distribution is desired for an accurate approximation of the nonlinear process. LFFC has successfully been applied to compensate for cogging in an industrial Linear Motor (Otten et al.[20]) and for compensation of (Coulomb) friction of a linear motor used in a flight simulator (Velthuis[19]). It has also been applied to the tracking control of the mobile robot described in the section Design of a Mobile Robot (Starrenburg et al.[21]).

The application to cogging compensation of a linear motor will be described in a little bit more detail. Such a motor is a commonly used element in assembly machines. Even with the best magnets and accurate assembly the error could not be made smaller than 100 $\mu$, with a PID controller in combination with nonlearning feed-forward control. The design goal was to improve the maximally achievable accuracy from 100 $\mu$ to less than 10 $\mu$. Figure 12.24 shows a picture of a linear motor.

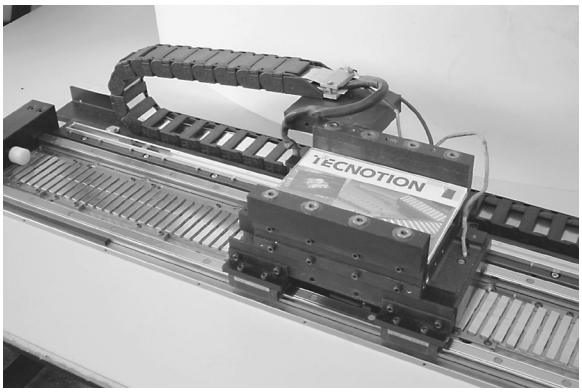



**FIGURE 12.24** Linear motor. The magnets that cause the cogging are clearly visible.

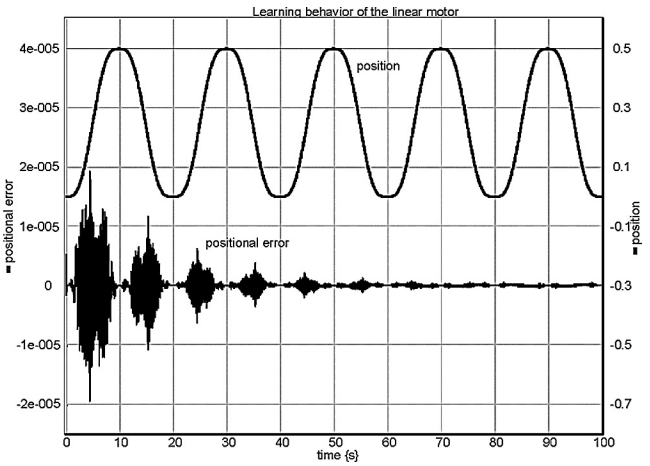



**FIGURE 12.25** Position and error signal during learning of the LFFC.

According to the structure of Figure 12.23 the linear motor is controlled by means of a PID controller, while a B-spline neural network is present to learn the inverse motor model, including the nonlinearity due to cogging. Cogging occurs in DC motors with permanent magnets. It causes more or less sinusoidal shaped forces that depend on the position of the translator with respect to the stator. If these forces really had a sinusoidal shape, they would be easy to compensate for by means of a feed-forward compensator. However, this would require magnets with completely similar magnetic properties and very accurate spacing of the magnets. An alternative is to design a controller that learns the disturbance pattern and compensates it by means of a learning feed-forward compensator. An additional advantage is that such a system can also be used to compensate for other nonlinear effects, such as friction. This has also been demonstrated in a part of a flight simulator (a control stick) where friction forces spoil the feeling of a realistic simulation especially at almost zero speed. Figure 12.25 shows that learning is almost completed after six training cycles.

Learning feed-forward control is an attractive method to compensate for nonlinearities that are present in mechatronic systems, such as cogging and friction. The use of B-spline neural networks results in fast convergence, relatively low computational effort, and a good generalizing ability. Because of recently obtained results with respect to the stability of such systems, robust control systems can be designed.

A mechatronic view on this design problem raises the question whether it is possible to use the same techniques to build a less expensive linear motor, when maximum accuracy is not the main goal of the design. It has been demonstrated that a motor constructed with less expensive components and less demanding assembly specifications but with LFFC can compete well with the more expensive construction. The accuracy can typically be improved by a factor 10.

## 12.5 Special Requirements of Mechatronics that Differentiate from "Classic" Systems and Control Design

The main difference between "ordinary controller design" and mechatronic system design is that the latter deals with the design of the system as a whole. This approach can be considered as optimization of all components of the system simultaneously, although there are no algorithms to do this automatically. In practice the problem is often split into smaller problems that can be optimized. After integration of all the partial solutions a suboptimal system is achieved that can be further optimized by retuning the different parts, taking into account the already available intermediate design of the overall system. In order to achieve optimization of the system as a whole, it is desired that the mechanical part, where power plays a role, and the information processing part (the controller) can be simulated and adjusted simultaneously. This requires that mechanical parameters like masses and compliances be available in simulations of the

controlled system. Examples have been given of modeling and simulation with 20-sim that allows for such an approach.

Mechatronic designers should constantly be aware of the fact that solutions can be found in different domains. Not every mechanical deficiency can easily be solved by control. A good mechanical design may be easier and cheaper to achieve. On the other hand, a good controller may be able to achieve the desired performance much easier and cheaper than a complex mechanical construction. In some cases the combination can even achieve performances that would never have been possible without a mechatronic design.

The same holds for the design of sensors. Each sensor could be fitted with a filter to remove noise from the measurements. But if several sensors are being combined, sensor fusion in a Kalman filter algorithm will benefit from the availability of the raw data.

Communication between all the designers involved and transparency of the design decisions in the various domains are essential for the success of a true mechatronic design.

## References

1. Matworks (2001). *The Mathworks: Developers of Matlab and Simulink,* www.mathworks.com
2. Coelingh, H.J., *Design Support for Motion Control Systems,* Ph.D. thesis, University of Twente, 2000, also www.rt.el.utwente.nl/clh/
3. Coelingh, H.J., de Vries, T.J.A., van Amerongen, J., Design support for motion control systems—application to the Philips fast component mounter, in *Mechatronics Forum 7th Int. Conf., Mechatronics 2000,* Atlanta, Ga, USA.
4. Controllab Products, *20-sim*, www.20sim.com
5. Broenink, J.F., *Computer-Aided Physical-Systems Modeling and Simulation: a Bond-Graph Approach,* Ph.D. thesis, University of Twente, 1990.
6. Dynasim, *Dymola,* www.dynasim.se/
7. Weustink, P.B.T., de Vries, T.J.A., Breedveld, P.C., *Object Oriented Modeling and Simulation of Mechatronic Systems with 20-sim 3.0,* in *Mechatronics 98,* J. Adolfson and J. Karlsén (Eds.), Elsevier Science, 1998.
8. De Vries, T.J.A., *Conceptual Design of Controlled Electro-Mechanical Systems,* Ph.D. thesis, University of Twente, 1994.
9. Breedveld, P.C., Fundamentals of bond graphs, in *IMACS Annals of Computing and Applied Mathematics, Vol. 3: Modelling and Simulation of Systems,* Basel, 1989, pp. 7–14.
10. Cellier, F.E., Elmqvist, H., Otter, M., Modeling from physical principles, in *The Control Handbook,* W.S. Levine (Ed.), CRC Press, pp. 99–108, 1996.
11. Gawthrop, P., Lorcan Smith, L., *Metamodelling: Bond Graphs and Dynamic Systems,* Prentice-Hall, NJ, 1996.
12. Van Amerongen, J., Modelling, simulation and controller design for mechatronic systems with 20-sim 3.0, in *Proc. 1st IFAC Conf. on Mechatronic Systems,* Darmstadt, Germany, September 2000, pp. 831–836.
13. Groenhuis, H., *A Design Tool for Electromechanical Servo Systems,* Ph.D. thesis, University of Twente, 1991.
14. Van Amerongen, J., Coelingh, H.J., de Vries, T.J.A., "Computer support for mechatronic control system design," *Robotics and Autonomous Systems,* vol. 30, no. 3, pp. 249–260, PII: SO921-8890 (99)00090-1, 2000.
15. Van Amerongen, J., van der Klugt, P.G.M., van Nauta Lemke, H.R., Rudder roll stabilization for ships, *Automatica,* vol. 26, no. 4, pp. 679–690.
16. Arimoto, S., A brief history of iterative learning control, in *Iterative Learning Control: Analysis, Design, Integration and Applications,* Kluwer Academic Publishers, pp. 3–7, 1988.
17. De Vries, T.J.A., Velthuis, W.J.R., van Amerongen, J., Learning feed-forward control: a survey and historical note, in *1st IFAC Conf. on Mechatronic Systems,* Darmstadt, Germany, September 2000, pp. 949–954.

18. De Kruif, Bas J., de Vries, T.J.A., On using a support vector machine in learning feed-forward control, in *Proc. 2001 IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics,* Como, Italy, 8–12 July, 2001.
19. Velthuis, W.J.*R., Learning Feed-Forward Control—Theory, Design and Applications,* Ph.D. thesis, University of Twente, 2000, also http://www.rt.el.utwente.nl/vts/
20. Otten, G., de Vries, T.J.A., van Amerongen, J., Rankers, A.M., Gaal, E., Linear motor motion control using a learning feedforward controller, *IEEE/ASME Transactions on Mechatronics,* vol. 2, no. 3, ISSN 1083-4435, pp. 179–187, 1997.
21. Starrenburg, J.G., van Luenen, W.T.C., Oelen, W., van Amerongen, J., Learning feed-forward controller for a mobile robot vehicle, *Control Engineering Practice,* vol. 4, no. 9, pp. 1221–1230, 1996.

# 13

# The Role of Modeling in Mechatronics Design




Jeffrey A. Jalkio
*University of St. Thomas*


If mechatronics design is more than just the combination of electronic, software, and mechanical design, the additional feature must lie in the ability of the mechatronic designer to optimize a design solution across these disparate fields. This requires a sufficient understanding of each of these fields to determine which portions of an engineering problem are best solved in each of these domains given the current state of technology. In turn, this requires the ability to model the problem and potential solutions using techniques that are domain independent or at least permit easy comparison of solutions and tools from different domains.

For example, the optical inspection system shown in Figure 13.1 depends on optical components in precise alignment, mechanical elements capable of precise motion, transducers for sensing and providing mechanical power, electrical systems to control motion and filter sensor signals, and software for image analysis and motion control. Only by dividing these tasks appropriately among electronics, mechanical components, and software can the system be optimized. This requires an understanding of all the system requirements and limitations as well as the capabilities of each component in the various domains. Modeling of requirements and systems is crucial in determining whether a proposed solution is acceptable as well as in documenting these determinations for future use. In this article we shall examine the varieties of models used at different points in the design process, the diverse roles of these models and their relative strengths and weaknesses in each of these roles, and finally the specific tradeoffs involved in choosing dynamic models for signals and systems analysis.

## 13.1 Modeling as Part of the Design Process

Models serve different purposes at different points in the design process; so to decide which modeling tools are most effectively employed in different phases we must examine the design process itself. Many descriptions of the design process are available that have been developed by researchers around the world.[1–3] Typically these descriptions serve to systematize the process to improve the productivity of

**FIGURE 13.1** An optical inspection system for printed circuit boards. (Used by permission © CyberOptics Corporation 2001, all rights reserved.)

designers or to describe techniques that provide improved product quality, lower cost, or other benefits. However, since our purpose is to examine the modeling needs of the design process, we can consider a simple model that distinguishes phases of the design process in terms of types of design activity rather than a more complex model that may be preferable for other purposes. For this purpose, we can consider a four-phase process consisting of requirements analysis, concept generation, analysis and selection, and detailed design. In the first phase of this process the designer focuses on analysis of the problem without considering possible solutions. In the second phase, conceptual solutions are generated with the hope that an acceptable solution can be found from these initial concepts via combination or modification of concepts or by variation of parameters present in one of the conceptual solutions. In the third phase, these concepts are evaluated and a design is chosen for implementation. The fourth phase consists of identifying design problems that need to be solved to implement the chosen concept and applying the design process to those smaller problems. We shall consider the activities of each of these phases in detail.

## Phase 1

The requirements analysis phase consists in obtaining a sufficient understanding of the problem to be solved. The difficulty of this process varies with the scale of the problem, the designers' familiarity with the problem domain, the variability of market needs, and the presence of hidden requirements that are poorly articulated in the initial problem statement. Depending on the nature of the design problem, the requirements identified in this phase may be the needs of a single customer, the common needs of a group of potential customers identified via a market survey, or societal needs identified by government regulations. Most design problems include some combination of these as well as internal requirements such as design guidelines and company policies. The key objective of this phase is to obtain enough detail to know when a design has solved the problem satisfactorily. Models in this phase serve primarily as communication and documentation tools since the primary problem in this phase is the clear communication and documentation of the criteria for design success. Examples of models that aid in this process include specification listings, use case diagrams, sequence diagrams, and context diagrams. Many of these modeling tools have now been standardized as parts of the Unified Modeling Language (UML), which is becoming increasingly important as an analysis tool.[4]

Use case diagrams model the interactions between a system and its users at a very high level of abstraction in terms of purpose of the interaction. Figure 13.2 gives an example of a use case diagram used to document the various operations required of a network printer. The use case diagram helps us avoid overlooking important but rare use cases such as maintenance. It is important to note that use case diagrams do not

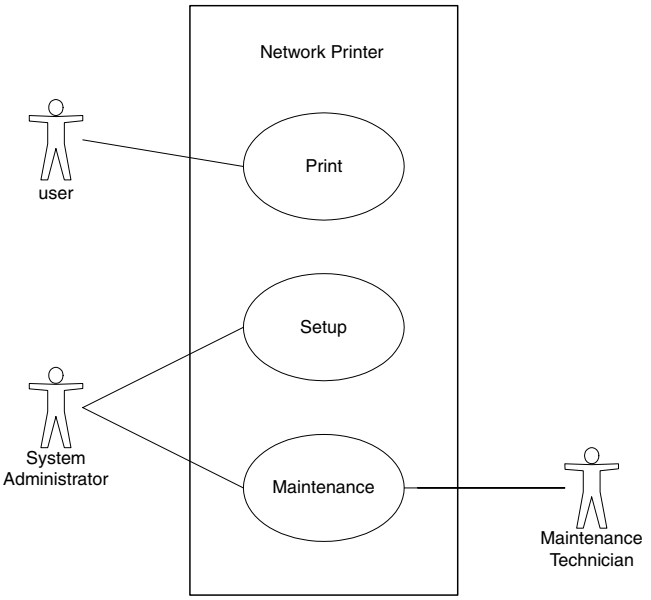



**FIGURE 13.2** Use case diagram for a network printer.

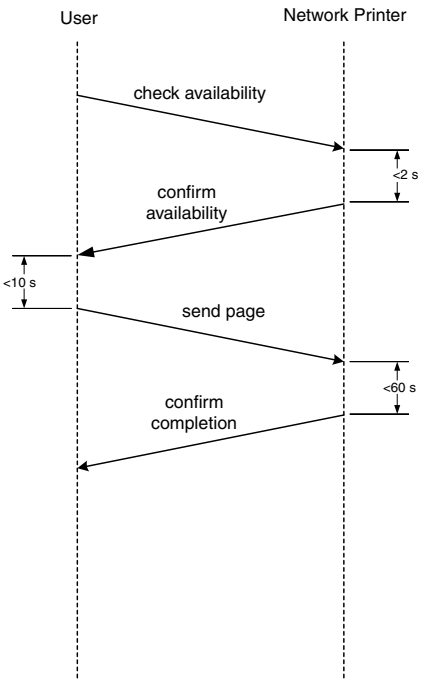



**FIGURE 13.3** A sequence diagram for network printer use.

provide information about the nature of the transaction, but rather documents its existence. It serves as a top level model of a system only.

Sequence diagrams can be drawn to describe the details of individual use cases in order to clarify the interactions that must occur, timing constraints, and interactions between the system and multiple elements of the environment. Figure 13.3 shows an example of a sequence diagram for the "print" use case of Figure 13.2.

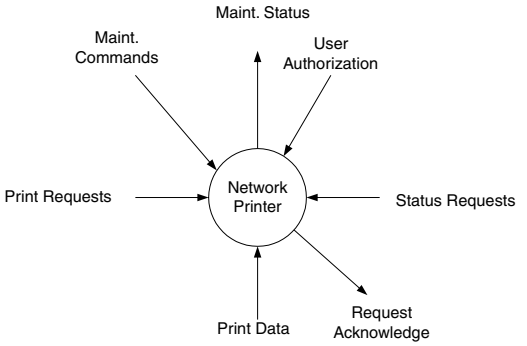



**FIGURE 13.4** Context diagram for network printer.

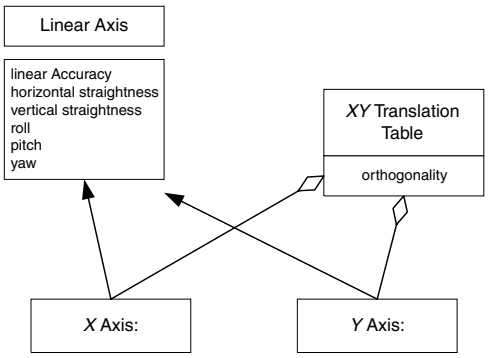



**FIGURE 13.5** Class diagram for linear translation stages.

Note that the sequence diagram documents a particular instance of a particular use case. The sequence diagram can show both the direction of information flow and timing requirements. In this regard the sequence diagram and the traditional timing diagram serve similar purposes.

Context diagrams do not capture timing elements but capture the types of information that flow between the system and its environment.[5] While use case diagrams focus attention on the scenarios of system use, context diagrams focus attention on information flows that must exist to enable those scenarios. Both serve useful and complementary purposes. Figure 13.4 shows a context diagram for an inspection system like the example mentioned above. Notice that the context diagram summarizes information flow shown in the various interaction diagrams for all use cases. Regulatory and safety requirements are often in the form of limits on the interactions the system may have with aspects of its environment. These requirements must all be captured for use in later design phases and communicated back to the source of the requirement for verification of accuracy. The context diagram is also used as the top level of a data flow diagram for a system if structured analysis is used.[6]

Just as the context diagram ties to structured analysis techniques, object oriented analysis techniques provides the class relationship diagram as a means of documenting the relationships between a system and its environment and between components of a system.[7] Class relationship diagrams show how a system is composed of subsystems, how components are similar to one another, and how they differ. For example, Figure 13.5 shows a class relationship diagram for a two-axis translation system consisting of two single axis subsystems. This diagram documents the existence of 13 error components that must be specified in the requirements phase. It does this by showing that the *X* and *Y* axis components are instances of a class of single axis translation stages, that each have six error components, and that they are components of a system that adds a single additional error. By documenting relationships such as composition and

inheritance, requirements and the interdependence of requirements can be represented in a compact and comprehensible way.

One key aspect of the requirements definition phase is the importance of defining requirements without specifying a preferred solution embodiment. Hence, modeling methods should be chosen to document these requirements and their intrinsic relationships without implying a particular solution.

## Phase 2

In the concept generation phase, our objective is to generate multiple design concepts that might satisfy the requirements identified in phase 1. Here we need modeling techniques that allow us to describe possible solutions with varying levels of detail dependent on the degree of detail needed to document the key elements of the concept. Since individual concepts generated in this phase may only satisfy some portions of the design requirements, it is critical that modeling at this point allow for partial descriptions of embodiments and for the easy combination of design concepts. For this reason, our models must clearly document the portions of the requirements satisfied as well as any unspecified parameters or additional requirements introduced in the concept. For some problems, block diagrams showing interconnections between components solving portions of the problem are a useful modeling tool at this stage. Figure 13.6 shows two possible block diagrams describing a given design concept. The first specifies a particular control algorithm, sensor, and actuator, while the second leaves these particulars unspecified and simply describes a closed loop controller. Depending on the situation either of these may be appropriate descriptions. The first provides details and is closer to a complete design while the second, being more generic, is easier to combine with other concepts to generate hybrid solutions.

Block diagrams are not the only modeling tool appropriate at this stage. For other problems, schematics showing arrangements of components or equations or psuedocode of proposed algorithms may be employed. As this phase continues, design concepts are often combined to form potential solutions to the overall design problem; therefore, it is useful if the model of each concept can contain descriptions of preconditions for its use, results, and other parameters that help a design team determine how to combine concepts. Similarly, once potential solutions are formed by concatenating concepts, it is often useful to clean up the final concept by combining features from different component concepts or eliminating overlapping features that are no longer needed.[8] This process is facilitated by modeling techniques that allow simultaneous modeling of mechanical, thermal, electrical, and software components and concepts. These techniques include the familiar linear graph and bond graph models.[9,10]

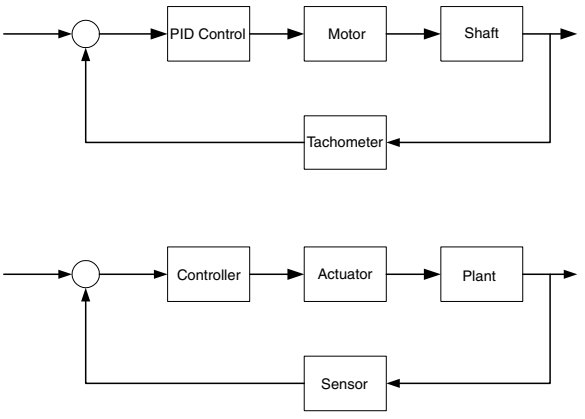



**FIGURE 13.6** Block diagrams at two levels of detail.

### Phase 3

In the third phase we evaluate potential solutions in terms of the problem requirements. This phase is eased if we have a model that allows us to compare problem requirements with design features. A particular design methodology may include various quality criteria in addition to those set by customer requirements and the regulatory environment, e.g., the design axioms of Suh prefer designs with minimum information content and which satisfy each requirement by independent features of the design.[11]

One approach to evaluation is to attempt to find numerical criteria for all requirements and evaluate solutions via minimization of an overall cost function (or equivalently, the maximization of an overall value function). This approach has proven effective in certain types of problems where requirements are amenable to quantification (for example, the optimal solution being the one that meets requirements with minimal weight or economic cost). Even in these cases, it can be difficult to determine the relative importance of diverse requirements in the formulation of the value function. For example, it may be desirable for a design to have minimal parts cost and minimal weight. An appropriate cost function might be

$$\Psi = \sqrt[n]{\left(\frac{c}{c_0}\right)^p + \left(\frac{w}{w_0}\right)^q}$$

where $c$ and $w$ are the cost and weight, respectively, and $c_0$ and $w_0$ are scaling factors indicating when the two factors are of equal importance, while the exponents $p$, $q$, and $n$ express the relative importance of minimizing the two factors. Clearly, there are many possible choices for these parameters and many other models that can be used. This highlights the difficulties of this method. An example of this difficulty in a relatively simple case is determining the cost of a component being out of tolerance. Models for this problem range from a step function of cost = 0 within tolerance and cost = C outside tolerance to Taguchi's quadratic cost function.[12] Needless to say, the "optimal" solution found is typically dependent on the cost function chosen. On the other hand, in some cases, the optimum found is not strongly dependent on precise choice of cost function and a satisfactory evaluation can be made without expending excessive effort to obtain an exact cost function. In fact, if there are relatively few design options from which to choose, it is possible to invert this problem. Rather than finding the design that minimizes a particular cost function, one can instead find the range of cost function parameters that result in any particular solution being optimal. It is often easy to determine which range of parameters is most realistic.

### Phase 4

The fourth design phase is detailed design, in which the entire process is repeated to resolve open design details for the individual components of the resulting design. This is in keeping with the design heuristic "allocate resources as long as the cost of not knowing exceeds the cost of finding out."[13] The process is inherently recursive, with each high level design decision producing a simpler design problem at a lower level of abstraction with simpler requirements. To accommodate this, we need to be able to model the design at multiple levels of abstraction and to allow the specification of interfaces between components. These models must allow us to define static as well as dynamic (behavioral) components of the interface.

In addition to this recursive aspect of design, there is of course iteration also possible, as dead ends are discovered in an otherwise promising chain of design decisions and as changes in the marketplace demand revisions of a product. To accommodate this iteration, design models should document not only decisions made, but also reasons for those decisions. Otherwise, the reasoning must be rediscovered in each iteration, or worse, may not be rediscovered until it is too late.

## 13.2 The Goals of Modeling

We have seen that models serve many purposes in mechatronics design. First, models serve to document concepts, assumptions, and requirements and allow the communication of these concepts to others in the design group and the various stakeholders of the design process. Second, models provide a hierarchical framework which allows division of labor in the design process and permits concurrent work on separate components.

Third, models can provide us with insights into the behavior of a system which might be obscured when we attempt to see the system in its full complexity. Fourth, they allow us to grasp similarities with other systems and use prior experience to help solve current problems. Fifth, models provide us a way of identifying and testing our ignorance of the actual behavior of the system by identifying unknown parameters and providing hypotheses worthy of testing. In this section, we shall consider these diverse purposes of modeling and examine the characteristics that cause a particular modeling methodology to be better or more poorly suited for each of these purposes.

## Documentation and Communication

The first purpose of a model that we shall consider is documentation. We often forget that aside from the various purposes of modeling in the design process, our models are often our best tools for communicating with our colleagues, our customers, and our successors. Every document generated in the design process is a model of some part of the system. To the extent that we can avoid double documentation of the same concept, we not only reduce workload and decrease time to market, but we also reduce the likelihood of inconsistencies in our documentation. How do our models serve as documentation? Consider a circuit diagram. It is not the circuit, but documentation of the circuit's design. It is also a model of the behavior we are seeking from the circuit. Nonidealities and component variations will cause the actual circuit to behave differently from this model, but the model still serves its purpose. Earlier in the design process, requirements documents also document the product, but in terms of requirements rather than the features that satisfy them.

Design documentation is important for at least four distinct groups. The engineering team itself needs communication among its members to ensure that effort is not wasted on elements that do not interoperate or that do not contribute to the overall product meeting requirements. Communication is required between the design team and the customer to ensure that requirements are correctly understood and implemented in the design. Management needs to understand the status of the project in terms of outstanding risks, ongoing expenses, and tradeoffs that would affect market size. Finally, future design teams can benefit from documentation of earlier design concepts, analyses, and decisions. This role of documentation as a communication with future workers in the field has been recognized as a key part of all scientific disciplines[14] but is often overlooked by engineers focused on meeting short-term deadlines.

What characteristics of a model lead to good documentation and clear communication? Clearly, the model must be understood by others. This requirement encourages the use of standardized modeling techniques whenever possible. It also encourages the use of modeling techniques that are interdisciplinary whenever feasible. When choosing models for documentation purposes, one must remember that each model highlights certain features of the design while hiding others. For example, the sequence diagram clearly documents communication between two objects while hiding details of how the objects generate outgoing messages or interpret incoming ones. If multiple models of the system are used to document the various aspects of the system, there should be some way to cross-reference items between models (e.g., determine that timing requirements documented in a sequence diagram model are met by the dynamics of a system described in a block diagram). Also, if the documentation is to be useful to future designers, it should be possible to cross reference design models based on a variety of criteria including system requirements and included components, so that future designs can benefit from current ones. Clearly, any modeling technique that serves some other purpose in design will also serve as documentation, but the model must be chosen so as to match the features one wishes to document.

## Hierarchical Framework

Real mechatronic systems include a large number of components that interact in many ways. Our models simplify these complex interfaces and describe products and processes in terms of simply interacting components with well-defined interfaces and behaviors. This simplification allows us to subdivide complex problems into a set of simpler problems whose solutions can be easily integrated. In complex mechatronic systems where we must consider components that interact in several energy domains, this

simplification is essential. When we draw a block diagram of a system we are dividing the system's behavior into a set of defined elements with modeled behavior and a known set of interactions between those elements, which is itself amenable to analysis. This role of modeling would lead us to select modeling techniques that allow us to divide the model into portions that can be independently tackled by independent engineers. For example, systems of linear differential equations may give us great insight into the fundamental modes of a system, but give us little insight on how to divide a design problem among members of a team. However, a block diagram, a class diagram, or a data flow diagram provides clear interfaces between elements that allow for subdivision.

## Insights

Regarding insights into the behavior of the system, different models provide different types of insights. Signal flow graphs of electromechanical systems show the presence of feedback loops that stabilize or destabilize the system. Differential equations provide insights regarding the time scales of various behaviors and the relative importance of various factors as well as providing estimates of the validity of simplifying assumptions. Similarly, sequence and timing diagrams can provide insight regarding the processing power required to meet timing requirements for various use cases. In each case the model provides insights into the problem or the characteristics of a proposed solution by bringing into focus certain aspects of the modeled system while hiding others from view.

## Analogies

Related to the insights a model can give us regarding the system under consideration are the insights that a model can provide in allowing us to see similarities to other systems that we or others have seen before. In this regard, modeling techniques that use analogies between various domains can be useful. However, analogies must be chosen with care. Consider the common analogy between electrical and mechanical quantities shown in the left half of Figure 13.7. In this analogy, velocity is analogous to voltage and force is analogous to current. However, as seen in the right half of Figure 13.7, the equations for the electrical system are unchanged in the dual system, in which current and voltage are exchanged and the roles of inductance and resistance are exchanged with capacitance and conductance, respectively. This dual system results in a different mechanical analogy, in which velocity is analogous to current and force is analogous to voltage. Each of these analogies can prove useful in moving design parameters from one energy domain to another, as the duality between the two electrical models can prove useful in circuit design. With any of these analogies, it must be remembered that real components deviate substantially from their idealized models and that the analogies do not strictly hold. This can be both a bane and a blessing, since a design that suffers from component nonidealities might be replaced by an analogous design from a domain with different but less detrimental nonidealities.

| Element | Electrical | Mechanical | Electrical | Mechanical |
|---|---|---|---|---|
| Capacitance | $i = C\frac{dv}{dt}$ | $f = M\frac{dv}{dt}$ | $v = \frac{1}{C}\int i dt$ | $f = k\int v dt$ |
| Inductance | $i = \frac{1}{L}\int v dt$ | $f = k\int v dt$ | $v = L\frac{di}{dt}$ | $f = M\frac{dv}{dt}$ |
| Resistance | $i = \frac{1}{R}v$ | | $v = Ri$ | $f = Bv$ |
| Conductance | $i = Gv$ | $f = Bv$ | $v = \frac{1}{G}i$ | |

**FIGURE 13.7** Electrical–mechanical analogies.

### Identification of Ignorance

The final role of modeling is to help us identify our ignorance. This takes two forms. First, the construction of the model often requires us to name parameters whose numerical value is unknown. This leads us to estimate the possible range of such parameters or to design experiments to determine the parameter's value. Identifying missing details is the first step in determining them. In this way models help us identify our ignorance of details of the problem or our proposed solution. Secondly, the predictions we make with the help of our models are testable. The adequacy of a model to describe system behavior can be determined by comparing actual system behavior with predictions based on that model. Differences between actual system performance and a model's predictions point to errors in the model that may be significant. By testing the validity of our models we identify aspects of the real system that are inadequately reproduced in our model or artifacts of the model that do not exist in the real system. These discrepancies may point to an inappropriate assumption or to a fundamental misunderstanding of the physical system. In either case, the source of these discrepancies should be understood in order to determine if our model is adequate for its intended purpose.

## 13.3 Modeling of Systems and Signals

In the area of systems and signals we typically deal with the modeling of system dynamics and information flows through the system and its environment. As can be seen from the discussion above, this is but a small part of the modeling needed for system design. However, there are a number of decisions that must be made in selecting a model for the behavior of a dynamic system. These include the choice between analytic and numerical modeling, the use of distributed or lumped parameters, the approach used to model random factors, and the choice between linear and nonlinear models. These issues are considered in the paragraphs below.

### Analytical vs. Numerical Models

In this area both analytical and numerical tools are available to support our efforts, so we have a number of decisions to make regarding the level of complexity of the models we choose to use for various tasks. Analytical tools provide insights into overall system behaviors and can allow us to make comparisons between dissimilar systems based on a similarity in their models. These analogous systems often prove useful in the concept generation phase of design because they allow us to bring to bear solutions to diverse problems to the solution of the problem at hand. However, as described above, these analogies usually break down upon closer inspection. In the case of analogies in dynamics, they are typically based on systems described by equivalent differential equations (e.g., electronic and mechanical oscillators). However, these linear differential equations are but simplifications that ignore nonlinearities and time-dependent behavior that may greatly affect system behavior. Therefore, when applying analogies one must always be careful to explicitly state assumptions made in the modeling process.

Numerical tools, on the other hand, do not yield the same insights of analytical tools, but provide other and more detailed insights into real system behavior and performance. Typically these tools become most useful later in the design process, both for tradeoff analysis and for detailed design work. The availability of these tools opens many doors to modeling options that have in the past been closed and allows us to consider options that have historically been considered unwieldy. For example, numerical solution of nonlinear differential equations allows modeling of systems that had previously been approached only with linear approximations.

### Partial vs. Ordinary Differential Equations

One modeling choice that must be made is whether we must model the dynamics of the system using ordinary or partial differential equations. Partial differential equations must be used whenever values of interest vary spatially as well as temporally. For example, if we are considering the design of a robot arm, we are free to use ordinary differential equations if we consider the arm to be rigid, partial differential

equations if we wish to describe the bending of a flexible arm in detail. However, what if we recognize a flexibility in the arm but do not need to know the details of its motion but only the effect of its flexure on the end effector? In this case we can make a lumped parameter model of the arm that summarizes its dynamics in terms of end effector motion and dynamic forces on the driver and end effector. When can this lumped parameter model be used? There are two restrictions. First, the details of the variation of behavior over space must be uninteresting to us. If we need to know these details (for example, to determine stress concentration, or temperature distribution) we cannot apply the simplified model. Secondly, the partial differential equation may not be amenable to forming a lumped form due to its complexity.

## Stochastic vs. Deterministic Models

One choice that must be made in the modeling of system behavior is how uncontrolled variability will be represented in the model.[15] The approach taken depends a great deal on the source and characteristics of the variability. For example, the values of some system parameters may not be precisely known, but may be constant over time. Others may vary slowly over time in unknown ways. In the first case, the sensitivity of the system to parameter value can be determined, using a variety of both analytical and numerical techniques. In the latter case, one must examine the speed of parameter variation to determine if the system can be analyzed in quasi-steady state or if the dynamics of the parameter variations are coupled with the dynamics of the system. For this purpose, it is often useful to write equations in dimensionless form where time and scale constants indicate critical speeds for coupling.

When the variation of parameters over time must be considered or when some inputs to a system have uncontrolled variability, we must choose whether to model the input as an arbitrary signal or a signal with known statistical parameters. In the first case, we have just another input and can analyze the system for sensitivity to this input. In the second case, we can characterize the system in terms of autocorrelation or equivalent spectral measures and use optimal filtering techniques to reduce uncertainty.[16] A key point in the statistical modeling of a system is to base the model on the known variability of the process rather than assuming additive, white, Gaussian noise at every turn.[17]

## Linear vs. Nonlinear

A wealth of techniques are available for the analysis of linear time invariant systems. Unfortunately, the world gives us nonlinear components. One advantage of modeling techniques that allow us to divide a complex system into simpler sub-components is that we can often isolate nonlinearities as individual elements and develop linear models for them. For small variations about a differentiable point on a nonlinear curve, we can always find a linear model, but the value of this model might be extremely limited depending on its accuracy over a reasonable range of input values. In the case of a discontinuous or nondifferentiable nonlinearity, there is a greater problem. A linear model can be found only when we are far from the discontinuity or when the discontinuity is small compared to the linear portion of the model.

However, nonlinearities also add essentially new behaviors to systems that are not possible in purely linear systems. For this reason alone, nonlinear models are needed in certain circumstances. As an example, for a linear system, stable oscillations are only theoretically possible for differential equations with purely imaginary eigenvalues. Positive real parts lead to growing oscillations, while negative real parts lead to damped oscillations. Clearly, the behavior of such a system is highly sensitive to system parameters. Small variations will either kill the oscillation or drive the system into nonlinear regions of operation. Furthermore, even for perfect oscillators, the amplitude of oscillations is unconstrained. However, a nonlinear system such as the Van der Pol equation:

$$\ddot{y} = -\omega^2 y + \alpha\left[1 - \left(\frac{y}{y_0}\right)^2\right]\dot{y}$$

includes a damping term that increases with the magnitude of the oscillation. For oscillations small compared to $y_0$ the oscillations grow, while larger oscillations are damped. This and other essentially

nonlinear behavior cannot be modeled adequately with linear models. Although once this behavior is recognized through the use of a nonlinear model, the resulting steady-state system (a stable oscillator) can be modeled for many purposes using its linear (although physically unrealizable) equivalent.

When using differential equations to model a system, we must be particularly careful in our choice of units of measure and in how we group parameters. For example, in the Van der Pol equation above, we can choose an alternative scale for $y$ such that $y_0$ is 1 and a time scale such that $\omega$ is 1. In this case we have

$$z'' = -z + \varepsilon(1 - z^2)z'$$

where $z = y/y_0$, $\tau = \omega t$, and $\varepsilon = \alpha/\omega$. This scaling provides a dimensionless equation as well as providing an indication of the system's natural time scale, the scale of $y$, and an indication that the size of $\alpha$ relative to $\omega$ is important. In general, we find that writing equations in dimensionless form provides three advantages. First, as mentioned above, the scaling factors provide a sense of time scale and magnitude of various features of the system's dynamic behavior. Second, the dimensionless parameters that result from combining physical system parameters provide a way of characterizing the behavior of a wide range of systems in terms of parameters that are easily mapped. Third, by putting the equation into dimensionless form, we are able to categorize it and recognize similar equations in different domains.

## References


1. Hubka, V. and Eder, W.E., *Engineering Design—General Procedural Model of Engineering Design,* Heuista, Zurich, 1992.
2. Pahl, G. and Beitz, W., *Engineering Design: A Systematic Approach,* Springer-Verlag, Berlin, 1988.
3. Dym C.L., *Engineering Design—A Synthesis of Views,* Cambridge University Press, New York, 1994.
4. Booch, G., Jacobson, I., Rumbaugh, J., and Rumbaugh, J., *The Unified Modeling Language User Guide,* Addison-Wesley, New York, 1998.
5. Douglass, B., *Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks and Patterns,* Addison-Wesley, New York, 1999.
6. Demarco, T., *Structured Analysis and System Specification,* Yourdon Press, NJ, 1978.
7. Coad, P. and Yourdon, E., *Object-Oriented Analysis,* Yourdon Press, NJ, 1990.
8. Glegg, G.L., *The Design of Design,* Cambridge University Press, London, 1969.
9. Shearer, Murphy, and Richardson, *Introduction to Dynamic Systems,* Addison-Wesley, Reading, MA, 1967.
10. Karnopp, D. and Rosenberg, R., *System Dynamics: A Unified Approach,* Wiley, New York, 1975.
11. Suh, N.P., *Principles of Design,* Oxford University Press, NY, 1990.
12. Taguchi, G. and Yokoyama, Y., *Taguchi Methods: Design of Experiments,* American Supplier Institute, Dearborn MI, 1994.
13. Koen, B.V., *Definition of the Engineering Method,* American Society for Engineering Education, Washington, 1985.
14. Lonergan, B., *Method in Theology,* University of Toronto, Toronto, 1990.
15. Zhou, K., *Essentials of Robust Control,* Prentice-Hall, NJ, 1998.
16. Papoulis, A., *Probability, Random Variables, and Stochastic Processes,* McGraw-Hill, New York, 1991.
17. Law, A. and Kelton, W., *Simulation, Modeling and Analysis,* McGraw-Hill, New York, 1991.

# 14

# Design Optimization of Mechatronic Systems




Tomas Brezina
*Technical University of Brno*

Ctirad Kratochvil
*Technical University of Brno*

Cestmir Ondrusek
*Technical University of Brno*


## 14.1 Introduction

Electromechanical systems form an integral part of mechanical and mechatronic systems. Their optimization is a necessary condition for a product to be competitive. In engineering practice, a large number of optimization and identification problems exist that could not be solved without the use of computers [5]. The present level of technological development is characterized by increasing the performance of machines with the production costs kept at a satisfactory level. The demands on the reliability and safety of operation of the designed machines are also considerable.

From practical experience we know that the dynamic properties of electromechanical systems have a considerable influence on their reliability and safety. On the other hand, the tendency to push the price of a machine down often leads to unfavorable dynamic properties that result in increased vibrations and noise during operation. Also, electrical properties dramatically deteriorate as the amount of active materials in a machine is reduced. The increased load leads to, among other things, excessive heat formation, which, in turn, has a negative effect on insulation, shortening the service life of a machine.

## 14.2 Optimization Methods

### Principles of Optimization

The properties of electromechanical systems can be described mathematically using physical quantities. The degree of these properties is then described using mathematically formulated objective (preference) functions. Structural parameters ranging between limit values given as satisfying secondary conditions are the independent variables of these functions. The particular form of the functions depends on the type of machine and its mathematical description. The solutions of a mathematically formulated optimization

problem together with optimization methods allow a considerable number of different design variants of a machine to be calculated in a relatively short time. They also make it possible to perform these calculations at production planning stages for a prototype to possess the qualities given by a chosen criteria function. In this way, the design of a machine is not only analyzed but also modified and reconstructed in terms of its electromechanical properties with the aim to improve these properties as much as possible (or optimize them).

From a physical point of view, these are actually problems that, to a certain degree, are inverse to those of calculus. A problem of calculus assumes a fixed, mathematically described model of a real machine to be used for deriving its resulting properties. In problems of calculus, we define properties and try to find out which parameters of a chosen mathematical model possess those properties. In problems of parametric optimization, we look for those parameters that, by a chosen preference function, provide the best properties. It is clear that problems of synthesis and optimization are much more sophisticated than those of calculus.

## Parametric Optimization

As the aim is to find the values of certain structural parameters of a machine, we shall deal with this notion in more detail. By a parametric optimization of electromechanical systems, we mean the process of finding those parameters of a mathematical description of the system (arranged in a vector $\bar{p}$) from a set $P$ of admissible parameters at which a suitably selected objective (preference) function $\psi(p)$ of these parameters reaches its extreme.

The objective function $\psi(p)$ quantifies the degree of the properties of an electromagnetic system that has to be made extreme (the parameters with the best degree of this property have to be chosen). When defining an admissible set $P$, we are guided by the structural possibilities of changes in individual parameters (variables), or we can introduce secondary criteria of the type "the degree of properties may not exceed given critical limits." The possibility of taking into consideration the structural changes of parameters leads to the so-called trivial (natural) constraints of the type $p_i^d \le p_i \le p_i^h$, where $p_i^d$ is the lower and $p_i^h$ the upper bound of the $i$th optimization variable. The introduction of secondary criteria leads to the definition of limiting functions $q_i$ of optimization variables for which we have $q_i^d \le q_i(p) \le q_i^h$, where $q_i^d$ is the lower and $q_i^h$ is the upper bound of the relevant function.

Thus, from the mathematical point of view, parametric optimization of electromechanical systems is formulated as the problem of finding a point $p$ in the admissible set $P$, at which the preference function $\psi$ reaches its global extreme value (maximum or minimum) with regard to $P$. The admissible set is generally described by $m$ inequalities defined by functions $q_j(p)$, where $j = 1, 2, \ldots, m$. If $P = R^s$, where $s$ is the number of variables to be optimized, we say that the optimization is unconditional. In all other cases we say that the optimization is conditional.

To solve the problem of optimizing the selected properties of a system, the following has to be done:

- a mathematical description has to be formulated,
- it has to be analyzed at the starting point,
- the desired form of the objective function $\psi$ has to be specified,
- the optimization variables have to be selected,
- the desired form of the constraining functions $q_j$ has to be specified,
- a suitable optimization method has to be selected,
- the resulting mathematically formulated optimization problem has to be solved, and
- using the mathematical model, the results have to be transformed back into the dynamic model (for dynamic problems only).

## General Aspects of the Optimization Process

If the aim of an optimization process is to optimize several properties that simultaneously affect the system (such as minimizing the size values while respecting the electrical properties), we obtain a multi-criteria objective function. The objective function then takes the form of a weighted sum of single-criterion

functions. Each of these functions generally assumes its local minima at different points of the optimization parameter spaces. This is the reason why a multi-criteria function can have a large number of shallow local minima or is insensitive to changes in the optimization parameters. Due to this fact, the selection of an optimization method is of great importance. The result is averaged in the sense that several criteria may participate simultaneously in a reduction of the multi-criteria function, while some other criteria may increase.

A more suitable method may be to select a single-criterion objective function, including all criteria in the constraints. Only the most significant criterion is chosen for the objective function to be specified in the subsequent process. All other criteria included in the constraints are kept within specified limits without being optimized. Thus, the results of an optimization process are dependent on the degree of reduction of the admissible set given by the inequality-type constraints.

Generally, we specify the constraints in a form similar to the objective function

$$q_i(p) = f_i(p) - f_i^h, \quad i = 1, 2, \ldots, m^* \tag{14.1}$$

Here $f_i$ are suitable functions of a vector variable and $f_i^h$ their maximum admissible values.

The selection of optimization variables is given by the sensitivity of the objective function to changes of relevant optimization variables. This sensitivity is described by the gradient vector of the objective function.

$$\text{grad } \psi(p) = \left[\frac{\delta\psi(p)}{\delta p_1}, \ldots, \frac{\delta\psi(p)}{\delta p_s}\right]^T \tag{14.2}$$

## Types of Optimization Methods

### Standard Optimization Methods

Most practical problems lead to nonlinear (transcendental) systems of equations. These may only be solved using numerical optimization methods. According to the order of the derivatives used in the application of a method, numerical methods of finding local minima of functions of several variables may be divided into:

1. zero-order methods (comparative)
   - methods of co-ordinate comparison
   - simplex methods
   - stochastic methods
2. first-order methods (gradient and quasi-gradient)
   - methods of associated directions
   - variable-metric methods
3. second-order method (Newton method)

#### *Stochastic Methods*

These methods consist of calculating the values of the objective function at a large number of selected points. The points are selected by such criteria that each point in the space has an equal probability of being selected. The best points are then determined by comparing the function values. From the outlined strategy, it follows that these methods lead to computing the function values at a large number of points, which may protract the calculation. On the other hand, we can more easily reach the global optimum of the function to be optimized. These methods also comprise the evolution methods since the first solution population is generated completely by random. The difference only consists in the strategy of selecting better solutions.

## Evolutional Optimization Methods

Since some problems are difficult to solve by standard numeric optimization methods, even if they converge to an acceptable optimum in a reasonable time, new approaches had to be developed. Therefore, a number of new methods have been designed based on the laws of natural genetics copying them in various degrees. These methods employ random generation of input parameters and so can be thought of as stochastic methods. In general, stochastic optimizing algorithms (including virtually all the evolutional algorithms) optimize using multi-parameter function with "wild" behavior, that is, with many minima or with an unknown gradient. Stochastic optimization methods are necessarily slower than heuristic approaches, which take advantage of the fact that they know the type and details of the function to be optimized. Unless the conditions for the global optimum are previously known, we can never be sure whether we have reached the global optimum to be able to terminate the optimization process. However, stochastic optimization methods also bring numerous benefits. They are generally very well specified and thus applicable virtually to any problem, and they can get out of the trap of a local minimum. The evolutional process of searching the space of potential solutions requires an equilibrium of two objectives:

- to find the nearest (mostly local) minimum as quickly as possible, and
- to search the space of all potential solutions in the optimum manner.

The methods differ in their orientation towards these two objectives and they can be roughly ordered in a sequence starting with methods tending to local minima to methods searching a large number of potential solutions:

1. Stochastic "hill climbing" algorithms,
2. Tabu search algorithms,
3. Simulated annealing algorithms, and
4. Genetic algorithms.

### *Hill Climbing Algorithm*

This is the simplest optimization algorithm being a variant of the gradient method "without gradient" where the direction of the steepest climb is determined by searching the neighborhood. This algorithm also has all the drawbacks of gradient methods, in that it is very likely to end up in a local extreme without reaching the global minimum. Here the starting solution is generated at random. For the currently designed solution, a certain neighborhood is generated using a finite set of transformations and the best minimum is chosen from this neighborhood. The local solution obtained in this way is then used as the center of a new neighborhood in which the optimization is repeated. This process is iterated a specified number of times. In the course of this process the subsequent best solutions are recorded to be finally used as the resulting minimum. The basic drawback of this algorithm is that, after a number of iterations, it may revert to a local minimum that has already been passed in a previous step (the problem of looping). This problem can be avoided by running the algorithm several times with different randomly generated initial values to eventually choose the best result achieved.

### *Tabu Search Algorithm*

At the end of the 1980s, Professor Fred Glover designed a new approach to solving the problem of finding the global minimum, which he called *tabu search*. At present, this method is among those used most frequently to solve combinatorial problems and problems of finding the global minimum. Based on the hill-climbing algorithm, it tries to eliminate the problem of looping. The hill-climbing algorithm is equipped with a so-called short-time memory, which, for a short previous interval of the algorithm history, remembers the inverse transformations to locally optimal solution transformations used to obtain the new centers in iterations. These inverse transformations are prohibited (tabu) when the new neighborhood is created for a given current solution. In this way, the looping caused by falling into the trap of a local minimum may substantially be reduced. A hill-climbing algorithm modified in this way systematically searches the entire area in which the global minimum of a function is to be found.

***Simulated Annealing Algorithm***

Apart from the stochastic methods and methods based on natural evolution, there is another possibility of simulating the evolution of systems based on the physical evolution of macroscopic systems. The annealing of a solid body in order to remove the internal stress is a simple example of this kind of evolution. For a physical interpretation of this process, consider a body that is heated until it reaches a high temperature. The temperature is then gradually lowered. The atoms of a body heated to a high temperature can easily overcome the local energetic barriers to reach equilibrium states. When the temperature is lowered, atoms are fixed in this state and the cooled off body is without internal stress.

This principle was used to design the method of simulated annealing. First, an initial temperature $T_{\max}$ is set, whose value is important for the method to be efficient. The simulated annealing algorithm then searches the space of all potential solutions in a strongly stochastic way, also accepting the states that correspond to solutions worse than the current one. This property of simulated annealing is a characteristic feature of this method and provides a way of escaping from a local minimum trap, thus allowing the search of another area of the entire solution space. However, as the annealing temperature $T$ is lowered, the probability of accepting worse states as well is diminishing. For small temperature values then, only solutions better than the current one are considered.

***Genetic Algorithm (GA)***

Genetic algorithms (GAs) are most frequently used to optimize the parameters of an unknown system whose mathematical description is either too complicated or unknown [5]. When applying a GA, it is mostly sufficient to know a function assigning a price to each individual in the population. This may be the error of the solution for randomly selected parameters during GA. Since a GA is looking for a maximum, the error, which, on the contrary, is being minimized, must be transformed into looking for a maximum. This may be done in several different ways: by subtracting the error from the maximum error occurring, by calculating the inverted value of the error, or by using another transforming function that approaches zero as the error approaches one. Increased attention should be paid to setting up the program implementing the pricing function since it consumes the most computing time compared with the other GA components.

Apart from general optimization problems, GAs are mostly applied to neural networks. Here the tendency is to employ GAs at two different levels. First, for finding suitable weights for a neural network and second, when optimizing the structure of a neural network, that is, when selecting the algorithm, the number of input neurons in the hidden layers, the number of hidden layers, etc. Using a genetic algorithm to optimize the parameters of another genetic algorithm (the size of the population, the number of crossbreedings, the extent of mutations, the frequency of mutations) is a very revolutionary idea (optimization of the computation time where the computation time is a pricing function of the GA). As far as applications of GAs to problems encountered in research of electric machines are concerned, GAs have been used to identify the parameters of the substitution diagram of an induction motor.

By way of conclusion, it may be added that genetic algorithms perform surprisingly well when all other algorithms fail, such as for incomplete problems where the computation time is an exponential or factorial function of the number of variables. There is no point in using GAs to optimize relatively simple functions or functions for which special algorithms exist for their description. Considering the necessity to calculate the function values for tens or hundreds of genetic chains in a population and the necessity to evaluate hundreds or even thousands of populations during a single run of the program, GAs are rather time-consuming.

Despite the positive results achieved by using GAs, it is clear that nature must use even more intricate and, at the same time, not very sophisticated methods. The GAs described above only correspond to very primitive examples observed in nature, particularly those related to asexual reproduction with a single chromosome. Since nature has taken billions of years to test its algorithms, it is highly efficient to further learn from it. It is interesting that it needs no mathematics to solve complicated problems of optimization. Nevertheless there are other optimization methods suitable to solve the problems of the design [2–4].

### Selection of a Suitable Optimization Method

The standard gradient method is still one of the most frequently used methods. Gradient methods or even the standard non-gradient methods (such as the simplex method) are not suitable if the finding of the global minimum is required of a function with many local minima. Mostly, these methods only reach an insignificant minimum close to the starting point (the initial solution) in which they are trapped. This deficiency is mostly removed by repeatedly selecting at random the initial solution of an optimization problem and taking the best result for the solution. The stochastic character of this process can only be seen in the random selection of the initial solution. The subsequent optimization algorithm then proceeds without any randomness. Then the evolutional optimization methods are thought of as stochastic ones despite their employing of a certain strategy when choosing the getter points. The following are the main differences between a genetic algorithm and the more frequently used gradient method:

- GA performs no gradient computation, which might be difficult and time consuming particularly for large systems, and
- GA works with randomly generated solutions and may converge more quickly to the global minimum.

To optimize the draft design of an induction motor, an optimization method was employed using a genetic algorithm. This method is described in more detail in the following chapter.

## 14.3 Optimum Design of Induction Motor (IM)

### IM Design Introduction

Actual design of an induction motor usually depends on the requirements of individual customers, who specifically define parameters which a designed machine should accomplish. In this way, with the same machine output, we can obtain different implementations that meet individual conditions more or less. It is possible to require a good quality of one parameter only with the deterioration of other parameters. We are going to deal with a design of motors of (0,6–200) kW outputs. Motors are designed for permanent load and with the project assignment the following input values are required:

Machine output $P_n$ [kW], voltage $U_{1n}$ [V], winding connection *Y*/*D*, number of poles 2**p** or rotation speed $n$ [min$^{-1}$], grid frequency $f$ [Hz], efficiency $\eta$[%], power factor cos $\varphi$, insulation class, IP implementation, and the shape of the machine.

We consider squirrel-cage motors in closed implementation with framework and cooling ribs. There is a cast aluminum rotor cage. For the design, data such as conductors and slots dimension or magnetic characteristics deducted from tables and graphs that are given by the standard or by the manufacturer's measurement, are needed. The actual design is a compromise between individual design parameters, so that a resulting machine would have the best possible operating characteristics with a perfect heat and material utilization. The actual motor design is described in the following section.

### Classical IM Design Evaluation

An induction motor design, when carried out manually, represents hundreds of calculations, which can last tens of hours even with an experienced constructor. As computers made their way into practically all branches of design and analysis, a series of programs which co-operate with a designer in an interactive fashion and speed up a calculation were created.

In spite of indisputable advantages of this design process, we have to realize that there is a remarkable quantity of various design implementations of the given motor which, more or less, achieve the required motor operating characteristics. This approximates the global minimum of an objective function, which evaluates the design quality.

Thus, the idea to create a program for searching the whole state space of all possible solutions and selecting such a variant, which is the most appropriate to an evaluating objective function (the required

**TABLE 14.1** Generated Parameters List and Setup of Their Limits

| Parameter Name | Symbol | Dimension | High Limit | Low Limit |
|---|---|---|---|---|
| Stator outside diameter | $D_e$ | mm | User optional | User optional |
| Stator inside diameter | $D$ | mm | User optional | User optional |
| Ideal iron length | $l_i$ | mm | User optional | User optional |
| Air gap induction | $B\delta$ | T | 0.5 | 1.0 |
| Stator slot filling | $k_{\text{dr1}}$ | — | 0.6 | 0.75 (0.8) |
| Air gap size | $\delta$ | mm | 0.2 | 0.4 |
| Stator current density | $\sigma_1$ | A mm$^{-2}$ | 3.0 | 15.0 |
| Rotor rod current density | $\sigma_t$ | A mm$^{-2}$ | 2.0 | 6.0 |
| Rotor ring current density | $\sigma_k$ | A mm$^{-2}$ | 2.0 | 4.0 |
| Teeth magnetic induction | $B_z$ | T | 1.6 | 2.0 |
| Slot number per pole and stator phase | $q_1$ | — | 2.0 | 5.0 |

motor characteristics) using some of the optimization methods, was developed. The stochastic evolutionary method genetic algorithm was selected, because it searches the whole state space of all possible solution in a best way.

## Description of a Solved Problem

### Generated Parameters

The values given in Table 14.1 are recommended only, and, for the motors of outputs below 200 kW, they are mostly limiting values. Varying the parameter values or substituting one parameter for another is possible only by intervention in the program source text. Diameter limits $D_e$ and $D$ from the input file are considered only in the case that a motor design without regard to standardized axis height is required. In the case that standardized axis height is entered, these limits are calculated. Limits of an ideal rotor length are appropriate to enter as narrow as possible for faster convergence to a limit. But this is not a required condition. Generally, the lower the range of individual parameters, the faster the convergence to a global minimum, and the number of local minimums is lower.

### Objective (Criterion) Function

It is not just the form of the objective function, but also the selection of optimized parameters that is important for good optimization results. Selected parameters must sufficiently describe a quality solution to the given problem. In the case of an induction motor design optimization task, the following parameters were selected:

| | |
|---|---|
| Motor volume | $V$ [dm$^3$] |
| Motor temperature rise | $\vartheta_n$ [K] |
| Motor nominal power factor | $\cos\varphi_n$ [–] |
| Motor nominal efficiency | $\eta_n$ [–] |
| Torque overload capacity | $m_{pn}$ [–] |

These parameters are most important for the quality of the design and describe the design sufficiently.

The total error is based on the relation given in Equation (14.1), a sum of individual partial errors of each controlled parameter. If we put more emphasis on some parameter, we increase a corresponding weight coefficient, thus achieving its improvement in the final design. At the same time, values of other parameters will decrease. Finding an optimal setup of gain coefficients is one of the most important and difficult problems. The term "optimal setup" means that the designed motor has the highest power factor, efficiency, and torque overload capacity values at the minimal volume and simultaneously does not exceed a permitted temperature rise for a selected insulation class. We have the relationship

$$\begin{aligned}\varepsilon(\text{GR}_i) \;=\; & \text{abs}(kV\cdot V) + \text{abs}(k\vartheta(0.89\,\vartheta_d - \vartheta_n)) + \text{abs}(k_{\cos\varphi}(1-\cos\varphi_n)) \\ & + \text{abs}(k\eta(1-\eta_n)) + \text{abs}(k_{mp}(m_p + 1 - m_{pn}))\end{aligned} \tag{14.1}$$

**TABLE 14.2** Input Values of 5.5 kW, 380 V Motor

| Quantity Name | Symbol | Dimension | Value |
|---|---|---|---|
| Nominal power output | $P_n$ | W | 5500 |
| Nominal voltage | $U_{1n}$ | V | 380 |
| Required power factor | $\cos\varphi$ | — | 0.81 |
| Required efficiency | $\eta$ | — | 0.86 |
| Grid frequency | $f$ | Hz | 50 |
| Motor axis height | $H$ | mm | 132 |
| Number of pole pairs | $p$ | — | 3 |
| Temperature class of insulation | $TT$ | — | F |
| Torque overload capacity | $m_p$ | — | 2 |

where $kV$ is the volume weight coefficient, $k\vartheta$ is the temperature factor weight coefficient, $k_{\cos\varphi}$ is the power factor weight coefficient, $k\eta$ is the efficiency weight coefficient, $k_{mp}$ is the torque overload capacity weight coefficient.

## Achieved Results

### 5.5 kW, 380 V Motor Design Description

During program development and tuning, an optimization was performed on the motor described in Table 14.2. In this section we describe the results and problems encountered in the optimization process. The symbols and quantities, which are not explained in detail, were either used in the previous text or are listed at the list of used quantities at the beginning of the document. The motor input parameters are given in Table 14.2.

### Other Results

It follows from the physical principle that optimized quantities are closely related. Increase of a gain of one quantity results in a disadvantage of the quantities. Based on the performed optimizations, it can be concluded that two kinds of motors exist, depending on the content of iron and copper:

1. Motor with prevailing iron content, high stator current density, good power factor, for a price of worse efficiency of the motor and with slightly worse torque overload capacity than the second motor.
2. Motor with high copper content and conversely low stator current density, good efficiency, and worse power factor value. Torque overload capacity is good.

The type of motor is determined based on the setting of gain coefficients. A sum of temperature and power factor errors on one side impacts the sum of volume and efficiency errors on the other side. The torque overload capacity can be good for both kinds of motors.

Results of individual optimizations are listed in Table 14.3, ordered by volume value from smallest to largest. Different varieties of motors were obtained, depending on values of gain coefficients. It is difficult to determine which solutions are good or bad, because the selections depend on actual customers' requirements. The solution that gives the best value of optimized quantity is marked in bold. Solution numbers 1, 5, 8, 23, and 25 can be considered successful from this perspective. The motor described above (solution no. 2) serves for depicting of the task. Previously-mentioned relations between individual quantities can be observed in Table 14.3, which lists the optimization results without limiting the generated parameter, thus using the requirements in Table 14.1.

Next, a motor optimization was performed with just one optimized parameter, when the gain of the other parameters was set equal to zero.

1. *Volume optimization*. In this case, the algorithm selected as the best solution motors with minimal dimensions, when parameters $D_e$, $D$, and $l_i$ were converging to minimum preset limits.
2. *Temperature optimization*. The algorithm reached first reached a local minimum with temperature at the maximum based on the required insulation class, and mostly stayed on this value.

**TABLE 14.3** Motor $P = 5.5$ kW, $U = 380$ V Solutions List, without Generated Parameters Limited

| Number | $V$ [dm$^3$] | $\vartheta$ [K] | cos$\varphi$ [–] | $\eta$ [–] | $m_p$ [–] | Directory |
|---|---|---|---|---|---|---|
| 1 | **3.96** | 88.1 | 0.798 | 0.834 | 1.72 | Motor1 |
| 2 | 4.20 | 86.9 | 0.818 | 0.843 | 1.90 | Motor2 |
| 3 | 4.31 | 74.9 | 0.787 | 0.865 | 1.77 | Motor3 |
| 4 | 4.32 | 88.8 | 0.836 | 0.817 | 1.78 | Motor4 |
| 5 | 4.33 | 75.1 | 0.690 | **0.973** | 1.07 | Motor5 |
| 6 | 4.50 | 89.0 | 0.836 | 0.834 | 1.79 | Motor6 |
| 7 | 4.51 | 86.8 | 0.818 | 0.818 | 1.93 | Motor7 |
| 8 | 4.54 | **90.0** | 0.884 | 0.812 | 1.98 | Motor8 |
| 9 | 4.56 | 84.6 | 0.857 | 0.816 | 1.74 | Motor9 |
| 10 | 4.58 | 86.5 | 0.836 | 0.817 | 1.77 | Motor10 |
| 11 | 4.63 | 68.2 | 0.792 | 0.858 | 2.10 | Motor11 |
| 12 | 4.69 | 88.4 | 0.862 | 0.808 | 1.80 | Motor12 |
| 13 | 4.70 | 73.4 | 0.845 | 0.830 | 2.25 | Motor13 |
| 14 | 4.73 | 61.0 | 0.799 | 0.871 | 1.90 | Motor14 |
| 15 | 4.78 | 78.1 | 0.853 | 0.858 | 1.67 | Motor15 |
| 16 | 4.78 | 71.0 | 0.767 | 0.870 | 1.80 | Motor16 |
| 17 | 4.81 | 70.6 | 0.703 | 0.934 | 1.28 | Motor17 |
| 18 | 4.97 | 54.5 | 0.804 | 0.883 | 1.90 | Motor18 |
| 19 | 5.08 | 55.5 | 0.762 | 0.877 | 2.20 | Motor19 |
| 20 | 5.12 | 88.2 | 0.879 | 0.806 | 2.05 | Motor20 |
| 21 | 5.96 | 44.0 | 0.784 | 0.870 | 2.55 | Motor21 |
| 22 | 6.35 | 42.4 | 0.803 | 0.882 | 2.69 | Motor22 |
| 23 | 6.40 | 87.5 | **0.887** | 0.853 | 2.27 | Motor23 |
| 24 | 6.57 | 59.0 | 0.747 | 0.956 | 1.16 | Motor24 |
| 25 | 7.05 | 42.3 | 0.793 | 0.865 | **3.00** | Motor25 |
| 26 | 7.39 | 52.9 | 0.714 | 0.986 | 1.03 | Motor26 |

3. *Power factor optimization*. An effort to achieve the first type of motor (see above discussion) with low copper content, high current density $\sigma_1$, worse value of efficiency. Torque overload capacity was good.
4. *Efficiency optimization*. The designed motor corresponded to the second type of motor (see above discussion) with prevailing copper content, low current density $\sigma_1$, and good efficiency, however, with worse power factor values. Torque overload capacity was good.
5. *Torque overload capacity optimization*. The motor is designed with high number of slots for pole and phase, resulting in gradual spread of conductors on the perimeter. The motor can have prevailing iron or copper content depending on a local solution, to which it converged. It can have good values of power factor and efficiency for a price of machine volume increase.

## 14.4 The Use of a Neuron Network for the Identification of the Parameters of a Mechanical Dynamic System

The basic step used to solve the dynamic tasks by means of any type of modeling is to create a set of important quantities that include both the quantities describing structure, conditions, and the interactions of technical objects and the quantities that characterize the consequences (i.e., their demonstration and behavior).

The methods of creating the mathematical models in drive systems, in general an interactive process, utilize

- the applications of well-known physical principles that describe the phenomena in drive systems (e.g., the second Newton's principle, Kirchhoff's laws, etc.), or
- the applications of the methods based on artificial intelligence algorithms (e.g., genetic algorithms [1] and artificial neuron networks [6, 7]).

The theories on which the methods of artificial intelligence are based replace the "standard" analytical and numerical methods when

- these are the only theories which can solve the problem,
- they exhibit better properties from the point of view of the problem solution (e.g., a better conditionality considering the changes of input values), and
- they allow the problems to be solved more effectively.

The last case is typical when we want to approach the real operational conditions as close as possible. From various methods of artificial intelligence, the stochastic evolution algorithms and the artificial neuron networks are being increasingly utilized in the field of the modelling of drive interactive systems. In the following section, two methods are shown that are applied to the problems of the analysis of dynamic properties in drive systems.

The solution of dynamics by means of the algorithms of artificial intelligence represents a solution of the following partial problems:

- Specifying the set of important (relevant) quantities,
- Selecting the theory which is suitable to solve the problems,
- Arranging the relations among relevant quantities so that they allow the selected algorithm of artificial intelligence to be used,
- Generating the training data, and the selection of the method of teaching, for example, in neuron networks, and
- Testing the quality of the results reached and their evaluation.

## Practical Application

Many identification methods are known and very often verified quite well in practical terms. The limit factors that make these procedures more difficult (e.g., the assumptions about system linearity, stationarity, and normality of the phenomena which occur in the systems, etc.) are also known. Hence, we have used an untraditional approach to the problem *f* identifying the dynamic properties in mechanical systems for which the use of neuron systems seems to be promising, and at the same time available for engineers' thinking.

### A Practical Application—Gearbox

The first case that was analyzed is a vehicle gearbox. Inputs (engine load), outputs (frequency-amplitude spectra of torsional oscillations), and the gearbox structure were known. The ilete information was related to the selected parameters of stiffness and damping in the drive. Due to variable operational conditions, the magnitude of damping may vary enough to significantly affect output characteristics. If many experimental results are evaluated, some typical failures can be identified and their possible occurrence can be anticipated from output files. We have used the data that were measured in the real system. The frequency-amplitude spectrum of torsional oscillations was measured at the gearbox shaft, which is seated on four bearings with five speed gears (Figure 14.1). Originally, the measurement was carried out to determine the resonance of frequency systems with the goal to reduce noise. The following parameters were set in the system:

Stiffness $K \in \{0.3, 12.0, \infty\}$,
Damping $B \in \{-, 0, 0.3, 12.0\}$.

The measurements were done with testing frequencies $f = 512$ Hz and $f = 1024$ Hz. To record significant oscillation harmonics, the excitation frequency was flexible in both cases:

1. from 2.5 to 14.0 Hz in steps of 0.5 Hz, and
2. from 14.0 to 40.0 Hz in steps of 1.0 Hz.

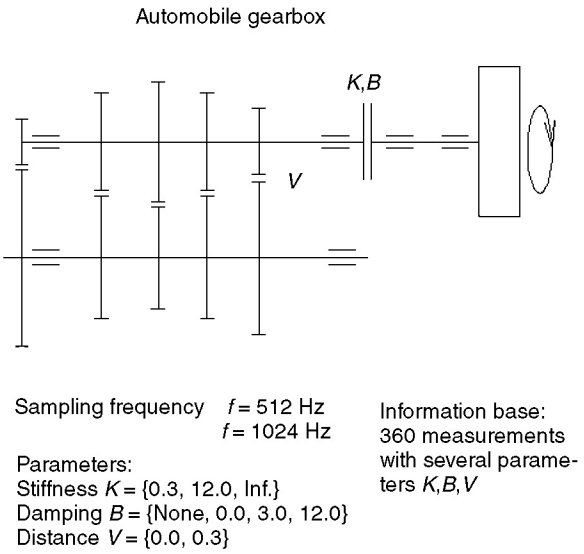



**FIGURE 14.1** Five speed gear.

**TABLE 14.4** Expected Natural and Excitation Frequencies for the Gearbox

| Table of Expected Frequencies | | [Hz] |
|---|---|---|
| Low frequencies | | up to 5.0 |
| Operational frequency (OF) (speed) | OF | 14.16 |
| | 2 × OF | 28.32 |
| | 3 × OF | 42.48 |
| Interharmonic frequency (IHF) | 0.5 × OF | 7.08 |
| | 1.5 × OF | 21.25 |
| | 2.5 × OF | 35.42 |
| Natural frequency (NF) | I.NF | 43.91 |
| | 0.5 × I.NF | 21.96 |
| | 2 × I.NF | 87.82 |
| | II.NF | 322.1 |
| | 0.5 × II.NF | 161.1 |
| | 2.0 × II.NF | 644.2 |
| Combination frequency (CF) | 2 × OF + 0.5 × I.NF | 50.28 |
| | OF + I.NF | 58.07 |
| | 2 × OF + I.NF | 72.23 |
| | 2 × OF + 2 × I.NF | 116.1 |
| Tooth frequency (TF) {TF = z.OF} | 1st speed gear | 184.1 |
| | 2nd speed gear | 325.7 |
| | 3rd speed gear | 424.9 |
| | 4th speed gear | 580.6 |

The spectrum always included 512 spectral lines. The measurement was repeated 360 times for different variations of the parameters $K$ and $B$ (the system adjustment). The values of natural and excitation frequencies, which are expected for the gearbox to be analyzed, are shown in Table 14.4.

## Task Definition

The task was originally defined in the following way: to estimate the corresponding magnitudes of parameters $K$ and $B$ (this means, to recognize the adjustment of parameters used in the mechanical dynamic system) by means of the artificial neuron network on the basis of the frequency-amplitude

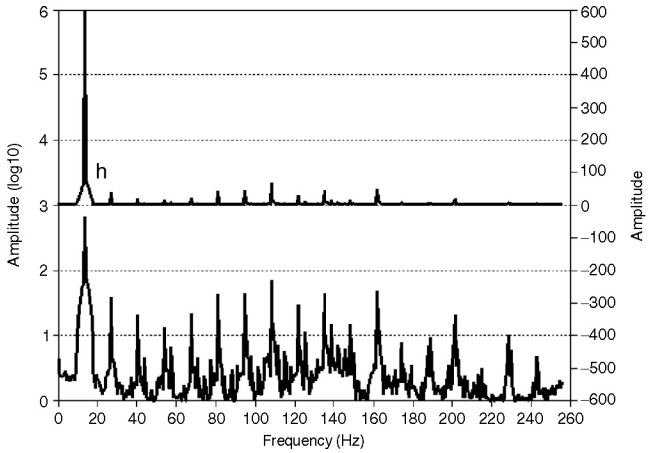



**FIGURE 14.2** Stimulus vectors normalization (top, before normalization; bottom, after normalization).

spectrum of torsional shaft oscillations in the given system. A multilayer perceptron with three layers (i.e., input, hidden, and output) is used as the configuration of the neuron network. Select the signal neuron functions as the linear, hidden, and output layers of the logistic function, $f(x) = 1/(1 + e^{-x})$.

According to this definition, this is to identify the system parameters on the basis of the measured frequency-amplitude spectra. However, the parameters are taken from discrete sets (and very low), and the task could be redefined as the "standard" task of the spectrum classification according to seven attributes (each attribute corresponded to one of the possible values of the parameters $K$ and $B$). The application of neuron networks to solve such a problem is more successful when compared to the solution of the original task.

The amplitudes of spectral lines were expressed in logarithm scale, and a reduction of spectral dynamics with an increase of their informative quality has been achieved. Considering the nonlinear nature of the activation neuron functions used, which extends beyond the saturation range for the input interval $\langle 0.5, 0.95 \rangle$, the network cannot respond well to stimulus vectors with a high range of the values in the individual components. This is illustrated in Figure 14.2. The input network layer was configured to 512 input neurons. The amplitude logarithmic value of one spectral line was entered into each input. The individual neurons in the output layer correspond to the classification attributes. Because there are seven attributes, seven neurons were configured in the output layer.

The only-hidden layer was set as the arithmetic mean of the number of input and output neurons. Two hundred sixty neurons were configured to the only-hidden layer, as illustrated in Figure 14.3. Each item corresponded to one measurement of the frequency spectrum (a stimulus vector) with a corresponding attribute vector (a vector of the required responses). The specific variation of the parameters was expressed by the required network response to two corresponding output neurons equal to 1, and the remaining output neurons equal to 0.

From the original 360 items, 36 items were randomly separated (10% of total) for the future tests. We ensured that the network tests would be carried out with the items that have not passed the training network process (the network was not trained to these situations). This is necessary to verify the generalization model properties. The training set was formed by the remaining 324 items. The sequential strategy of teaching was used, i.e., the items from the training set were used in the teaching process with the fixed sequence (cyclic passages through the training set). Taking into consideration the size of the neuron network to be configured, the method of feedback moment propagation has been selected as the teaching method that exploits only the information up to the first-order inclusively (the values of a special function—a teacher and his gradient), and it has not used Hessian or its estimate, which, in this case, would be very demanding.

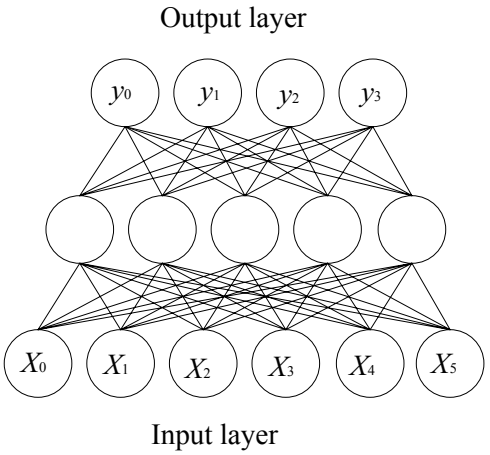



**FIGURE 14.3** Network configuration (hidden layer contains neurons without labels).

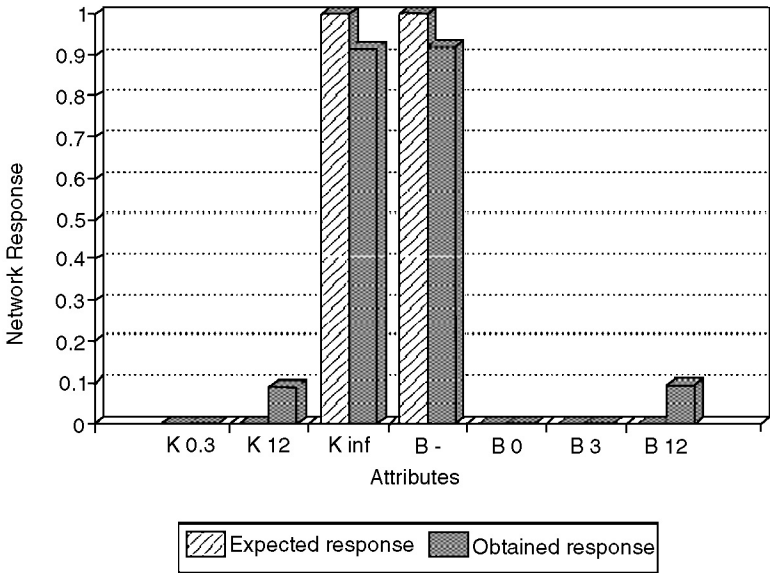



**FIGURE 14.4** Successful response of neural net.

## Results

The neuron model of the mechanical system has manifested a high rate of success during the verification by test sets. The network was taught with random selections of the test items. During testing of the individual models, the responses of the network were successful in 85–95% of all cases (see Figures 14.4 and 14.5). Moreover, the estimate of the values $K$ and $B$ that correspond to the parameters is available within a couple of seconds for the frequency spectrum in the active mode. However, it is possible that a model with higher quality will be achieved if special optimizing techniques are used in the future.

In summary, the neuron model of the mechanical system described above can be assessed as usable in practical terms.

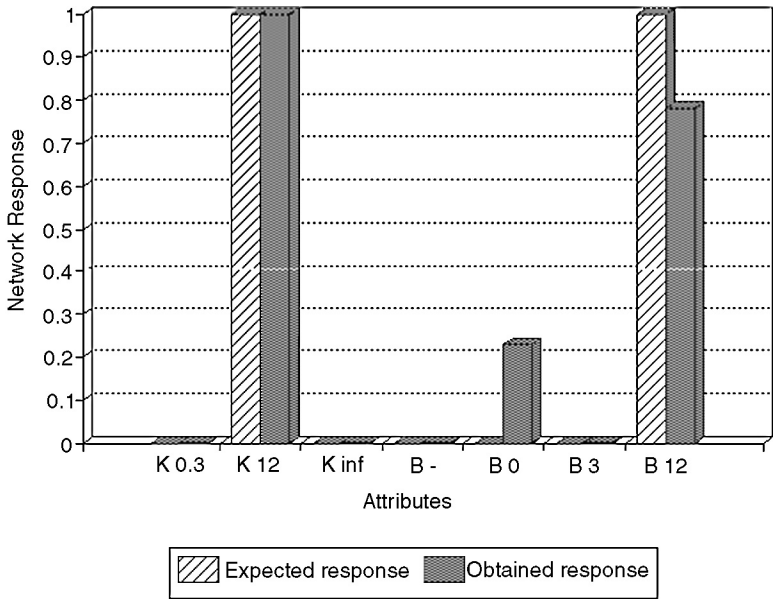



**FIGURE 14.5** Failure response of neural net.

## References

1. Goldberg, D., *Genetic Algorithms in Searching, Optimisation and Machine Learning,* Reading, MA: Addison-Wesley, 1989.
2. Glover, F., Lagunai, M., Marti, R., Fundamentals of scatter search and path relinking, *Control and Cybernetics,* pp. 653–684, 2000.
3. Glover, F., *Scatter Search and Star-Paths—Beyond the Genetic Metaphor,* pp. 125–137, New York: Springer-Verlag, September 1995.
4. Glover, F., Kelly, J.P., Langunai, M., Genetic algorithm and tabu search—hybrids for optimization, *Computers and Operations Research,* pp. 111–134, January 1995.
5. Lee, J., Hajela, P., Parallel genetic algorithm implementation in multidisciplinary rotor blade design, *Journal of Aircraft,* Vol. 33, No.5, pp. 962–969, September–October 1996.
6. Hagan, M.T., Demuth, H., Beale, M., *Neural Network Design,* Boston: PWS Publishing, 1996.
7. Kosko, B., *Neural Networks and Fuzzy Systems,* Englewood Cliffs, NJ: Prentice-Hall, 1992.
8. Ye, X., Loh, N., Dynamic system identification using recurrent radial basis function network, *Neural Networks Theory, Technology, and Applications,* New York: IEEE Technology Update Series, 1996.

# 15

# Introduction to Computers and Logic Systems

Kevin Craig
*Rensselaer Polytechnic Institute*

Fred Stolfi
*Rensselaer Polytechnic Institute*



## 15.1 Introduction: The Mechatronic Use of Computers

Mechatronics is the synergistic combination of mechanical engineering, electronics, control systems, and computers. The key element in mechatronics is the integration of these areas through the design process. Synergism and integration in design set a mechatronic system apart from a traditional, multidisciplinary system. In a mechatronic system, computer, electronic, and control technology allow changes in design philosophy, which lead to better performance at lower cost: accuracy and speed from controls, efficiency and reliability from electronics, and functionality and flexibility from computers. Automotive engine-control systems are a good example. Here a multitude of sensors measure various temperatures, pressures, flow rates, rotary speeds, and chemical composition and send this information to a microcomputer. The computer integrates all this data with preprogrammed engine models and control laws and sends commands to various valves, actuators, fuel injectors, and ignition systems so as to manage the engine's operation for an optimum combination of acceleration, fuel economy, and pollution emissions.

In mechatronics, balance is paramount. The essential characteristic of a mechatronics engineer and the key to success in mechatronics design is a balance between two sets of skills:

- Modeling (physical and mathematical), analysis (closed-form and numerical simulation), and control design (analog and digital) of dynamic physical systems
- Experimental validation of models and analysis and understanding the key issues in hardware implementation of designs

In mechatronic systems, computers play a variety of roles. First, computers are used to model, analyze, and simulate mechatronic systems and mechatronic system components and, as such, are useful for control design. Second, computers, as part of measurement systems, are used to measure the performance

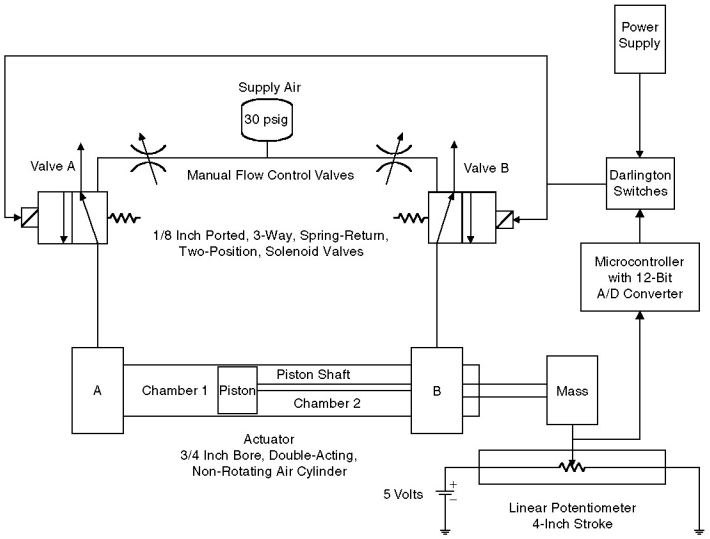



**FIGURE 15.1** Pneumatic servomechanism.

of mechatronic systems, to determine the value of component parameters, and to experimentally validate models. Finally, computers or microcomputers form the central component in digital control systems for mechatronic designs. Thus, computers play an essential role in the two essential characteristics of the mechatronics balance and comprise a key component to mechatronic system designs. This is illustrated by the following example.

Consider the schematic of a pneumatic servomechanism, a computer-controlled, closed-loop positioning system, shown in Figure 15.1. Pneumatic servomechanisms have the advantages of low cost, high power-to-weight ratio, ease of maintenance, cleanliness, and a readily-available and cheap power source. However, the disadvantages are high, nonlinear friction forces, deadband due to stiction, and dead time due to the compressibility of air. The design goal is to implement a fast, accurate, and inexpensive pneumatic-actuator system using inexpensive on/off solenoid valves, rather than expensive continuously-variable servo valves. To accomplish this task, one must completely understand the physical system, develop a physical model on which to base analysis and design, and experimentally determine and/or validate model parameters. One must then develop a mathematical model of the system, analyze the system, and compare the results of the analysis to experimental measurements to validate the model. One must then design a closed-loop position control system utilizing on/off, modified on/off, or pulse-width modulated control. Finally, one must implement the control system and experimentally validate its predicted performance.

A MatLab/Simulink model of this system is shown in Figure 15.2. The mathematical model is highly nonlinear, as are the various control schemes. A computer numerical simulation is needed to understand the behavior of the system and the various control schemes. A data acquisition system is needed to take measurements of the various system inputs and outputs and validate the numerical simulation. And, a computer (a microcontroller in this case) is needed for the real-time implementation of the various control schemes. There are a variety of computer numerical simulation tools available, some requiring the detailed mathematical model while others enable virtual prototyping where the various system components are assembled on the computer screen with the component mathematical models given hidden in the background. There are also a variety of computer platforms on which to run the control algorithm, e.g., high-end PC using a DSP board and a real-time control-code generator; a microcontroller programmable in C or Basic with an analog-to-digital (A/D) converter and numerous digital input/output (I/O) ports; and a microchip implementation needed for product development.

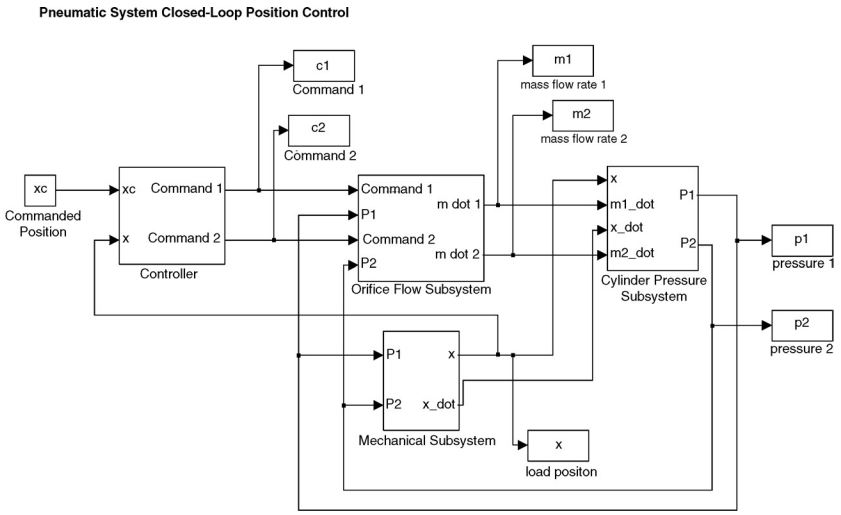



**FIGURE 15.2** MatLab/Simulink model of the pneumatic servomechanism.

## 15.2 Mechatronics and Computer Modeling and Simulation

In design, balance is the key to success, i.e., balance between theory and practice and balance between modeling/analysis skills and hardware-implementation/measurement skills. Figure 15.3 illustrates the steps in a dynamic system investigation, which is the process that would be utilized to design a mechatronic system. The distinction between physical modeling and mathematical modeling is emphasized, as is the importance of both analytical and numerical solutions to the model equations. To generate a physical model, approximations must be made to the actual physical system. Small effects are neglected. The influence of the environment is ignored. Elements are assumed to be lumped instead of distributed. The dynamics are assumed to be linear. Parameters are assumed to be constant. Noise and uncertainty is ignored. These approximations have a direct influence on the mathematical model. Neglecting small effects limits the number of equations. Environmental independence reduces the complexity of the equations. Other approximations result in linear ordinary differential equations with constant coefficients. Neglecting uncertainty avoids the use of statistics in the model. In most cases, a design consideration is to develop the simplest model which adequately depicts the complexity of the system dynamics.

The predicted dynamic behavior of the model is only half the story, for these results, without experimental verification, are at best questionable, and at worst useless. Comparing the predicted dynamic behavior with the actual measured dynamic behavior is the key step in the dynamic system investigation process.

The steps in the dynamic system investigation process should be applied not only when an actual physical system exists (as in reverse engineering) and one desires to understand and predict its behavior, but also when the physical system is a concept in the design process that needs to be analyzed and evaluated. After recognizing a need for a new product or service, one uses past experience (personal and vicarious), awareness of existing hardware, understanding of physical laws, and creativity to generate design concepts. *The importance of modeling and analysis in the design process has never been more important than in this situation.* These design concepts can no longer be evaluated by the build-and-test approach because it is too costly and time consuming. Validating the predicted dynamic behavior in this case, when no actual physical system exists, becomes even more dependent on one's past hardware and experimental experience.

In physical modeling, one first specifies the physical system to be studied along with the system boundaries, input variables, and output variables. In modeling dynamic systems, we use engineering judgment and simplifying assumptions to develop a physical model. The complexity of the physical model depends on the particular need, e.g., system design iteration, control system design, control design

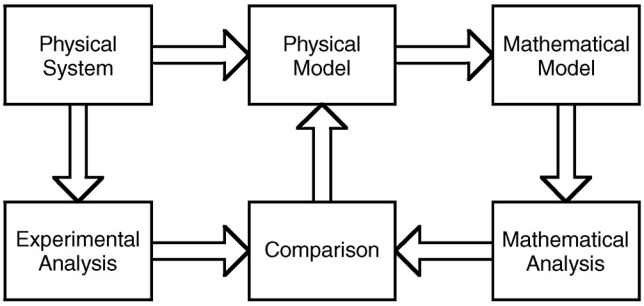



**FIGURE 15.3** Dynamic system investigation process.

verification, physical understanding. The intelligent use of simple physical models requires that we have some understanding of what we are missing when we choose the simpler model over the more complex model. The astuteness with which these approximations are made at the onset of an investigation is the very crux of engineering analysis. A variety of engineering models may be developed based on the particular need. Always ask the question: "Why am I modeling the physical system and what is the range of operation that I wish my model to be valid for?" If the need is system-design iteration or control-system design, then a "*design model*" is needed, i.e., a physical model whose mathematical model is a linear ordinary differential equation with constant coefficients and, therefore, useful with a broad, highly-developed assortment of linear design techniques. If the need is design verification before actual hardware implementation, then a "*truth model*" is needed, i.e., a physical model that is as close to reality as possible; with nonlinear simulation tools available, almost any mathematical model can now be simulated. Iterations can then be performed using, as a starting point, the results of the work performed with the design model. Models only need to be valid for the particular range of operation of interest; low-order models then can often represent very complex, higher-order models very effectively. In practice, you may need a hierarchy of models of varying complexity: a very detailed truth model for final performance evaluation before hardware implementation, several less complex truth models for use in evaluating particular effects, and one or more design models.

## 15.3 Mechatronics, Computers, and Measurement Systems

Measurement systems or data acquisition systems may be used for a variety of purposes, and a computer plays an integral role in each.

1. *Monitoring of Processes and Operations.* Certain applications of measuring instruments may be characterized as having essentially a monitoring function, e.g., thermometers, barometers, and water, gas, and electric meters.
2. *Control of Processes and Operations.* An instrument can serve as a component of a control system. To control any variable in a feedback control system, it is first necessary to measure it. A single control system may require information from many measuring instruments, e.g., industrial machine and process controllers, aircraft control systems.
3. *Experimental Engineering Analysis.* In solving engineering problems, two general methods are available: theoretical and experimental. Many problems require the application of both methods and theory and experiment should be thought of as complementing each other. Further, all models need validation, and measurement systems offer a means to collect the data required for model validation.

The distinction among monitoring, control, and analysis functions is not clear-cut; the category that a given application may fit may depend somewhat on the engineer's point of view and the apparent looseness of the classifications should not cause any difficulty. Rather it should be realized that computers,

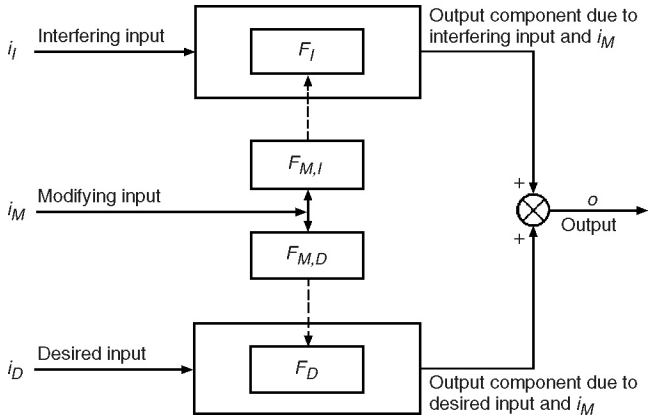



**FIGURE 15.4** Input–output configuration of a measurement system.

as general purpose processing elements, can serve many functions in the processing of measured parameters from mechatronic systems and that these processing functions can be related to or unrelated to the modeling and control of such systems. Special purpose digital signal processing electronics are also used in measurement systems. High-speed digital signal processors (DSPs), for example, are used to collect input and output signals in the determination of transfer functions for mechatronic systems. The high speed allows the processing of simultaneous samples of the input and output for minimal phase error. The primary application for DSPs in mechatronic systems, however, is real-time control, discussed below.

Figure 15.4 is the input–output configuration of a measurement system. Input quantities are classified into three categories:

1. *Desired Inputs*. These are quantities that the instrument is specifically intended to measure.
2. *Interfering Inputs*. These are quantities to which the instrument is unintentionally sensitive. $F_D$ and $F_I$ are input–output relations, i.e., the mathematical operations necessary to obtain the output from the input. They represent different concepts depending on the particular input–output characteristic being described, e.g., a constant, a mathematical function, a differential equation, a statistical distribution function.
3. *Modifying Inputs*. These are quantities that cause a change in the input–output relations for the desired and interfering inputs, i.e., they cause a change in $F_D$ and/or $F_I$. $F_{M,I}$ and $F_{M,D}$ represent the specific manner in which $i_M$ affects $F_I$ and $F_D$, respectively.

There are several methods for canceling or reducing the effects of spurious inputs. One method which relies upon computer processing of the signals is the method of calculated output corrections. This method requires one to measure or estimate the magnitudes of the interfering and/or modifying inputs and to know quantitatively how they affect the output. Then it is possible to calculate corrections, which may be added to or subtracted from the indicated output so as to leave (ideally) only that component associated with the desired input. Since many measurement systems today can afford to include a computer to carry out various functions, if sensors for the spurious inputs are provided, the computer can implement the method of calculated output corrections on an automatic basis.

## 15.4 Mechatronics and the Real-Time Use of Computers

We turn to the field of closed-loop control using a digital computer as the controller. Several comments are in order. First, a mechatronic system typically involves continuous variables. Elements rotate or translate in space. Fluids or gasses flow. Heat or energy is transferred. Computers are, by their nature, digital elements. Variables are represented in a computer by discrete values or simply by collections of zeroes and ones. For a computer to be used as the controller for a mechatronic system, therefore, the

continuous variables must be converted to discrete variables for processing and then back again to continuous variables. This might seem obvious. What is not so apparent is that the computer algorithm forms an inherent separation between the processing of the signals and the signals themselves, which is not true of other mechatronic system components. Even if digital logic elements are used (as discussed in this chapter) the signals are converted to discrete form, but the flow of information is still continuous through the elements. When a computer is used for the control element, this information flow is broken and buried in the computer algorithm. As an example, computer algorithms sometimes mimic continuous proportional-integral-derivative (PID) control laws. When the execution of this algorithm is analyzed, even if the effects of sampling and quantization are included, it is assumed that the signals are processed just as if they were being determined by continuous processing elements. In reality, if the computer code is examined at the machine level (i.e., not in the high level language in which it may be written), it would bear very little resemblance to a differential equation representation of the PID algorithm. This has practical implications both for modeling the exact operation of the computer as a control element and for validating that the computer code actually produces the desired response to signals.

Other issues are involved when the mechatronic system controller is implemented in software. Software execution is often asynchronous to the other time constants in the system (i.e., the software execution and system response are often not synchronized). Software can be made synchronous by syncing it to the sampler period, but this typically limits performance and is difficult if the computer is to be used for other tasks than control. Once a computer is contained as an element in a mechatronic system, there is a tendency to use some of the processing power to provide additional functionality or ease of use for the product. This additional code can affect, sometimes adversely, the operation of the real time controller execution. Testing of the code and safety of the code are also issues. The engineer has to determine that his system operates deterministically and safely for all possible combinations of input signals and for all possible states in the execution of the algorithm. For real-time systems, execution order for the code is often not predictable since it can be dependent on the particular combination of input signals. Simplicity of the code, providing for testability of the code, using established software quality assurance practices, and developing extensive documentation are ways to achieve system determinism and safety. Often, a hardware interlock, that is, a safety system utilizing electronic or mechanical hardware, is often included in software controlled systems.

Code operation has to be further verified as the code is modified and as the code is reused for systems other than that for which it was developed. Unlike other controllers, computer code is portable, but this requires more thought for its possible reuse. Using standard software packages, standard processors, modular code, and commercial real-time environments increases the possibility for reuse.

Besides the issues inherent in using computer code as the controller, there are issues involved whenever a digital processing component is incorporated into a mechatronic system. Further, there are considerations that must be taken into account whenever digital signals are processed. Figure 15.5 shows a configuration useful for this discussion. The computer is important, but the computer "component" of many mechatronic machines and processes is often not the critical system element in terms of either technical or economic factors. Rather, components external to the computer, the actuators and sensors, the sampling system, and the anti-aliasing filter are more often the limiting factors in the system design.

Since both continuous (analog) and digital signals exist in computer-controlled systems, the signals in such a system can be classified as shown in the table below.

| Signal Classification | Discrete in Time | Continuous in Time |
|---|---|---|
| Discrete in amplitude | D-D (digital) | D-C |
| Continuous in amplitude | C-D | C-C (analog) |

For analog signals, the precise value of the quantity (voltage, rotation angle, etc.) carrying the information is significant, meaning that the specific waveform of input and output signals is of vital importance. Conversely, digital signals are binary (on/off) in nature, and variations in numerical value are associated

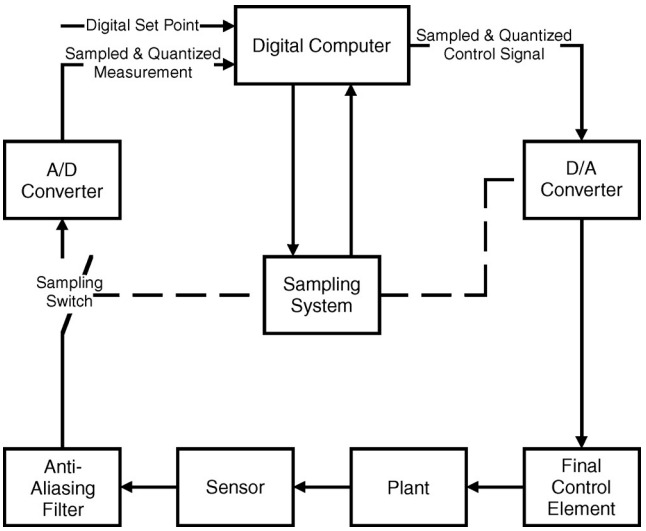



**FIGURE 15.5** General computer-control configuration.

with changes in the logical state (true/false) of some combination of switches, for example, +2 V to +5 V represents ON state, 0 V to +0.8 V represents OFF state.

In digital devices, it is simply the presence (logical 1) or absence (logical 0) of a voltage within some wide range that matters; the precise value of the signal is of no consequence. Digital devices are therefore very tolerant of noise voltages and need not be individually very accurate, even though the overall system can be extremely accurate. When combined analog/digital systems are used, the digital portions need not limit system accuracy; these limitations generally are associated with analog portions and/or the analog-to-digital (A/D) conversion devices. Since most mechatronic systems are analog in nature, it is necessary to have both A/D converters and digital-to-analog (D/A) converters, which serve as translators that enable the computer to communicate with the outside analog world.

In most cases, the sensor and the final control element are analog devices, requiring, respectively, A/D and D/A conversion at the computer input and output. There are, of course, exceptions, e.g., stepper motor and optical encoder. In most cases, however, the sensors can be thought of as providing analog voltage output and the final control element will accept an analog voltage input.

The current trend toward using dedicated, computer-based, and often decentralized (distributed) digital control systems in mechatronic applications can be rationalized in terms of the major advantages of digital control:

- Digital control is less susceptible to noise or parameter variation in instrumentation because data can be represented, generated, transmitted, and processed as binary words, with bits possessing two identifiable states.
- Very high accuracy and speed are possible through digital processing. However, hardware implementation is usually faster than software implementation. Determining the time required to develop a system in software is notoriously difficult to estimate.
- Digital control can handle repetitive tasks extremely well, through programming.
- Complex control laws and signal conditioning methods that might be impractical to implement using analog devices can be programmed. Very sophisticated algorithms can be implemented digitally.
- High product reliability can be achieved by minimizing analog hardware components and through decentralization using dedicated computers for various control tasks.

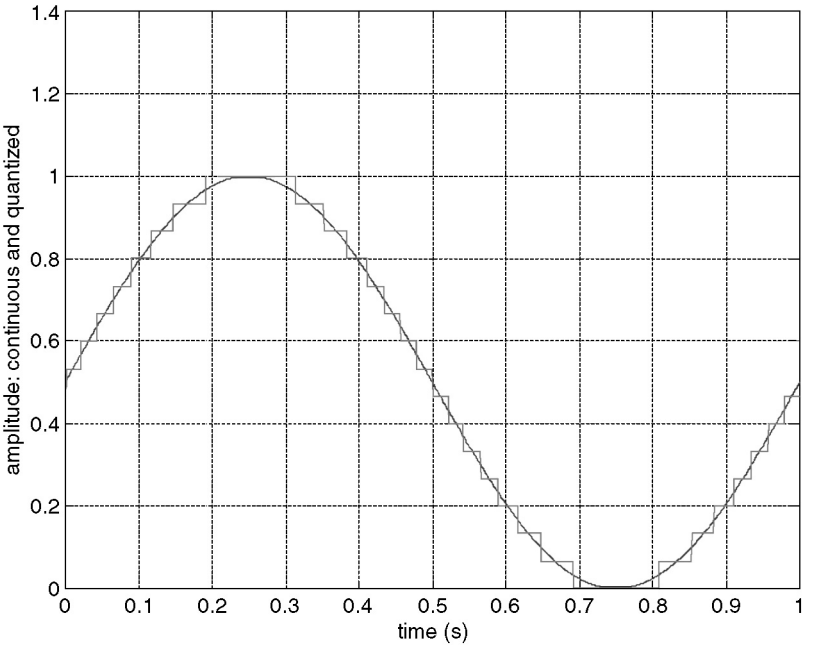



**FIGURE 15.6** Simulation of a continuous and 4-bit quantized signal.

- Digital systems are more easily "programmed" and offer the ability to time-share a single processing unit among a number of different functions.
- Large amounts of data can be stored using compact high-density data storage methods.
- Data can be stored or maintained for very long periods of time without drift and without being affected by adverse environmental conditions. Digital control has easy and fast data retrieval capabilities.
- Fast data transmission is possible over long distances without introducing dynamic delays, as in analog systems.
- Digital processing uses low operational voltages (e.g., 0–12 V DC).
- Digital control has low overall component cost.

Further, from the standpoint of the mechatronic product, the inclusion of a computer means that additional system functions can be provided. The user can select from a range of operations. Additional features can be included. A user interface providing indications of operation can be added with minimal cost.

In a real sense, some of the problems of analysis and design of digital control systems (beyond the issues associated with software) are concerned with taking into account the effects of the sampling period, $T$, and the quantization size, $q$. If both $T$ and $q$ are extremely small (i.e., sampling frequency 50 or more times the system bandwidth with a 32-bit word size), digital signals are nearly continuous, and continuous methods of analysis and design can be used. It is most important to understand the *effects of all sample rates*, fast and slow, and the *effects of quantization* for large and small word sizes. Lower cost computers are typically slower and have a smaller word size. Figure 15.6 shows the effects of having too few quantization levels, i.e., too small a word size. The signal that will be processed by the controller has large errors over the original analog signal.

Figure 15.7 shows the effects of sampling. It is worthy to note that the *single most important impact* of implementing a control system digitally is often the delay associated with the D/A converter, i.e., $T/2$. This pure delay results in a substantial phase shift in the closed-loop feedback system and often limits the control operation.

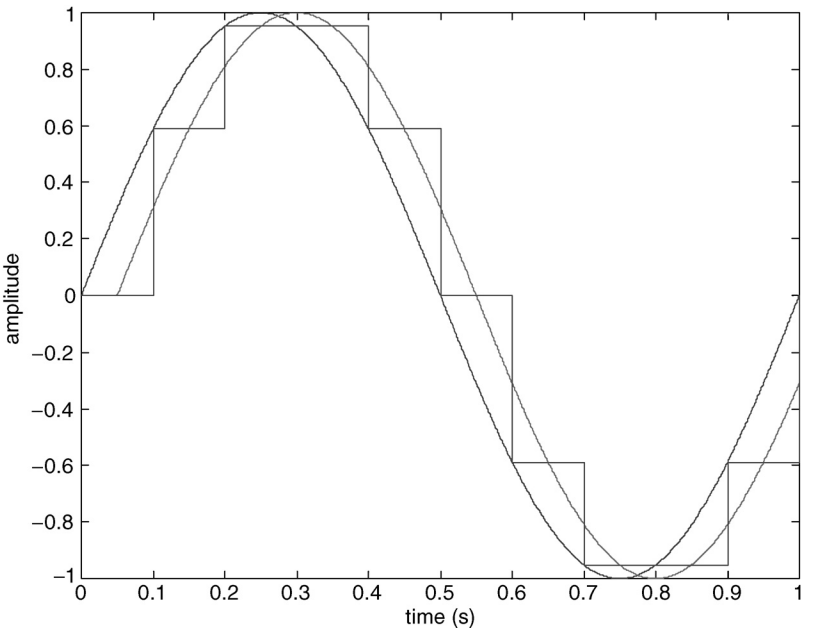



**FIGURE 15.7** Continuous and D/A converter output.

In a feedback system, the analog signal coming from the sensor contains useful information related to controllable disturbances (relatively low frequency), but also may often include higher frequency "noise" due to uncontrollable disturbances (too fast for control system correction), measurement noise, and stray electrical pickup. Such noise signals cause difficulties in analog systems and low-pass filtering is often needed to allow good control performance. The phase shift from this filter also adversely affects control system stability.

Finally, in digital systems, a phenomenon called *aliasing* introduces some new aspects to the area of noise problems. If a signal containing high frequencies is sampled too infrequently, the output signal of the sampler contains low-frequency ("aliased") components not present in the signal before sampling. This is illustrated in Figure 15.8. If the higher frequency signal is sampled too infrequently, the result will be exactly the same values as the low frequency signal. From the standpoint of the controller, there is no way for the system to distinguish which signal is present. If we base our control actions on these false low-frequency components, they will, of course, result in poor control. The theoretical absolute minimum sampling rate to prevent aliasing is two samples per cycle; however, in practice, rates of about 10 are more commonly used. A high-frequency signal, inadequately sampled, can produce a reconstructed function of a much lower frequency, which cannot be distinguished from that produced by adequate sampling of a low-frequency function.

In all of the above, the word computer was used for the digital processing element. In electronics literature, a distinction is usually drawn between a microprocessor, microcomputer, DSP, and computer. There is no standard for what each of these terms can mean, but some insight can be gained by examining Figure 15.9, which is a general block diagram for a computer. All computers have a means of getting input, a means of generating output, a means of controlling the flow of signals and operations, memory for data storage, and an arithmetic logic unit (ALU) which executes the instructions. The ALU and control elements are often called the central processing unit (CPU). Small computers, which just contain a CPU, are often called microprocessors. Memory for these computers is often attached to the microprocessor but in distinct electronic packages. Input and output to the microprocessor is often handled by electronics called peripherals. If the memory is included in the same package, the computer is called either a microcomputer or computer depending on its physical size. CPU and memory on a single electronics chip is

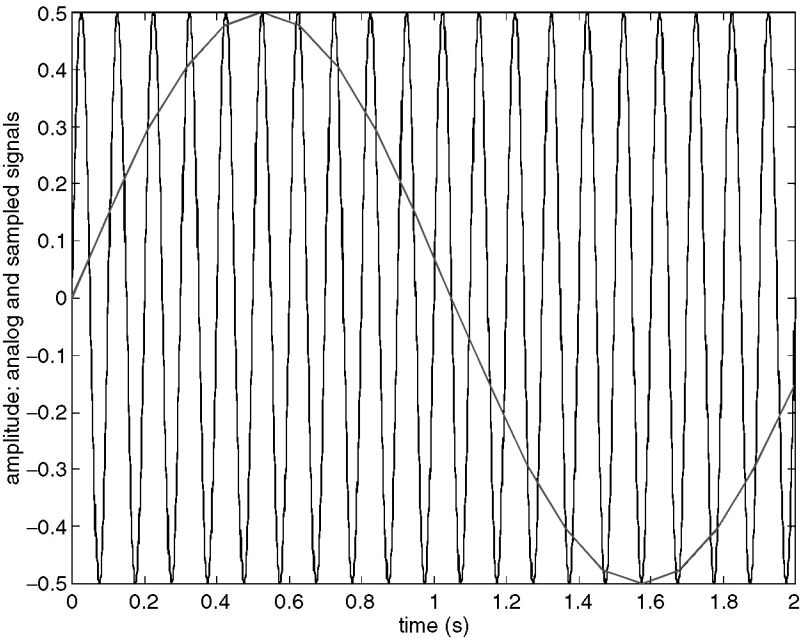



**FIGURE 15.8** Simulation of continuous and sampled signal: aliasing.

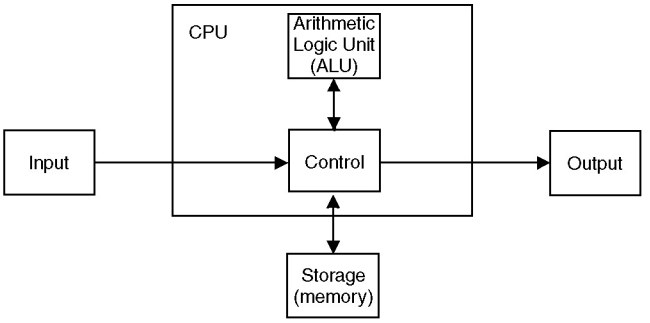



**FIGURE 15.9** Elements of a computer.

often called a microcomputer. The reader should be aware that a single electronics package can contain many "chips," which are connected by fine wires within the package. The overall package is still called a chip. Finally, if the A/D and D/A functions are provided in the same package, the computer is often called a DSP. However, these functions can also be contained in something which is called a microcomputer. DSPs are also computers which have a special instruction in the ALU called a multiply-accumulate (MAC) instruction even if the A/D and D/A are not present. Digital signal processing algorithms often involve MAC instructions and a computer, which can execute this instruction very effectively (in one instruction cycle of the computer), and are often called DSPs. To further complicate the situation, electronic devices called application specific integrated circuits (ASICs) exist. These devices can be custom made to perform a specific operation (such as a PID algorithm). ASICs can contain a CPU or memory or peripheral functions or even a MAC cell as part of its makeup. If the reader is thoroughly confused by this explanation, he probably has the proper grasp of the situation. However, he should be aware that diagrams like the one shown in Figure 15.9 often accompany the electronic component so the internal capabilities can be determined.

Before leaving computers, one final point will be made. Memory in a computer can often be divided between program space and data space, as shown in Figure 15.10. This representation is meant to be pictorial rather than to define a specific computer architecture. In a von Neumann architecture, for

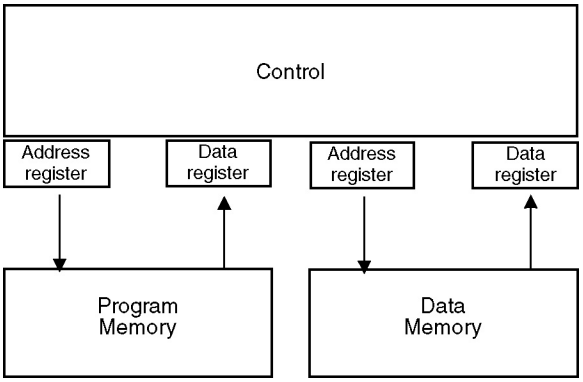



**FIGURE 15.10** Computer memory organization.

example, the program memory and data memory share the same space and information busses. Whereas in a Harvard architecture, program memory and data memory are distinct (looking more like the figure). In either case, for a mechatronic system, one can think of the program (in program memory) as the set of instructions which tells the CPU how to manipulate data (in data memory) to produce an output. This view should emphasize the earlier point that the flow of signals in a mechatronic system becomes confused if a computer is to be used for real-time control.

Because of the low cost of modern microcomputers, the use of logic elements as discrete components in a mechatronic system has diminished. Microcomputers are often programmed to perform logic functions, which has the advantage that the operation can be altered in software rather than requiring electronic hardware changes. In analyzing this logic, of course, any of the traditional methods can be employed. The logic can be minimized via Karnaugh maps, for example. The only difference lies in the implementation of the algorithm. ASICs are also used to implement logic functions.

## 15.5 The Synergy of Mechatronics

As stated at the beginning of this section, mechatronics is the synergistic combination of mechanical engineering, electronics, control systems, and computers and the key element in mechatronics is the integration of these areas through the design process. The use of computers and logic elements as components in mechatronic systems will produce successful designs only if this synergy is achieved. The system must be designed as a system. Computers should never be an add-on component included when the design is complete. When computers are synergistically incorporated in the system, the power of the mechatronics approach to design is realized.

# 16

# System Interfaces




M.J. Tordon
*The University of New South Wales*

J. Katupitiya
*The University of New South Wales*


This chapter deals with asynchronous serial interfaces described by interface standards RS-232, RS-422, and RS-485 and with the general-purpose parallel interface bus described by IEEE-488 standard. The chapter also provides background information, terminology and parameters, which are important in the design of system interfaces for mechatronic systems.

## 16.1 Background

Modern mechatronic systems comprise a number of subsystems, which rely heavily on digital data communications. Different levels of complexity of these systems means that the requirement for data communications range from a simple communication between two devices to systems with a large number of subsystems, where each subsystem communicates directly or indirectly with other subsystems using a communication network. Depending on the proximity of subsystems, different requirements are placed on data communication channels, the physical implementation of channels, and interfaces between these devices. Figure 16.1 shows a schematic diagram of a simple data communication system connecting two devices.

A data source creates the data to be transmitted to the destination system and may convert the data into a specific form. The originating system usually does not create the data in a form suitable for transmission over transmission lines. This is left to the transmitter, which transforms the data into a signal suitable for transmission over a specific type of transmission line. The transmission line is generally implemented using electrical wiring but can involve a variety of physical medium including radio frequency, infrared, and sound signals. A transmission line provides a physical medium connecting the two systems. A receiver accepts the


**FIGURE 16.1** A schematic diagram of a simple data communication system.

signal and converts it to the data form suitable to be passed onto the destination system. A data destination processes the data in order to recover the original information. From the previous information, it follows that even in the case of a simple data communication system a number of subsystems is involved in the communication task.

## Terminology and Definitions

**Interface:** The common boundary between two subsystems is called an interface, and as can be seen in Figure 16.1, a number of interfaces can be involved even in a simple communication system.

**Bit:** The simplest form of data is one bit, which can take one of the two values 0 or 1, and hence is called binary data. All information in modern digital computers is stored in binary form.

**Byte:** A fixed number of bits (usually 8), which can be treated by a computer as a unit.

**Character:** Historically, the information is expressed in terms of characters. A character is a member of a character set. An example of a character set is the set of characters in the English language.

**Character code:** Individual characters from the selected character set are encoded in digital computers as binary numbers. One of the most widely used character set codes is the American Standard Code for Information Interchange (ASCII).

## Serial vs. Parallel

The basic unit of information to be transferred between subsystems is usually a character. For short distances, multiple parallel lines can be used to carry out simultaneous transmission of all the bits of a character. For the transmission of data over long distances, the cost of multiple data lines is often prohibitive and it is normal to serialize the data so that it can be passed over a single data path as a stream of bits.

## Bit Rate vs. Baud Rate

The speed of data transmission is usually expressed as a number of data bits transmitted per second and is called an effective bit rate with a unit bps. Larger units like kbps (1,000 bps) and Mbps (1,000,000 bps) are commonly used. The baud rate is a signaling rate and is expressed as a number of times per second that the signal transmitted over a data transmission line changes state. For systems using only two states, the signaling bit rate is equivalent to the baud rate. Distinction should be made between the effective data transmission bit rate and the signaling bit rate. In asynchronous serial communications, the effective data transmission bit rate can be significantly lower than the signaling bit rate because of the inclusion of start, stop, and parity bits. To maximize the transmission speed over a serial line, modern communication systems use signals with more than two states, thus achieving higher signaling bit rates. For example, if the transmission signal uses 16 states, then the signaling bit rate is four times higher than the baud rate. The terms baud rate, signaling bit rate, and effective data transmission rate are often used interchangeably which leads to confusion.

## Synchronous vs. Asynchronous

For both parallel and serial interface, the problem of synchronization must be solved. The communication over a transmission line can be done either in synchronous or asynchronous communication mode. In synchronous communication mode, the transmission of data is synchronized with a clock; thus, the transmission is occurring at regular time intervals. Since the data transmission takes place at fixed times, the

completion of data transfer does not have to be acknowledged. In asynchronous transmission mode, the two systems are using clocks, that are not synchronized and may run at frequencies slightly out of step. Thus, for asynchronous systems, data validation requires a separate scheme called handshaking.

## Data Flow-Control

Another problem in asynchronous communication systems is the speed of data processing. If one system is significantly slower in processing the data, a flow-control must be implemented to avoid data loss. Data flow-control may require additional handshaking. Similar problems may arise in multitasking systems in which, due to other tasks, the system is unable to handle incoming data during the period of high workload.

## Handshaking

In order to ensure efficient transmission of data without errors, the sending system will use a separate signal to indicate that valid data has been presented to the interface. Because the instant at which the receiving device can process the data is not known, the sending device must wait for an acknowledgment signal before presenting new data to the interface. The handshaking can be implemented in either hardware or software.

## Communication Protocol

Operation of a communication system is governed by a set of rules which must ensure reliable data transfer without errors and data loss. Such a set of rules is called a communication protocol.

## Error Handling

Data transmitted over a communication line are subjected to noise and can thus be corrupted. Since it is essential to maintain the integrity of data, a number of different schemes for error detection have been developed. The simplest remedy after error detection is retransmission of the corrupted data. More sophisticated communication protocols can involve complex error correction schemes implemented at protocol level.

## Simplex, Half-Duplex, Full-Duplex

In its simplest form, communication can be established with a single pair of wires. The data transmission mode, in which data can pass in one direction only, is called simplex or unidirectional channel. In most applications it is required that the communication takes place in both directions. If the cost of the data transmission line is high, it can be arranged that signals can pass in either direction over a single transmission line using additional circuitry on both ends of the transmission line but only in one direction at a time. This type of data communication mode is called half-duplex. Additional handshaking is required to implement the time sharing of the transmission line.

If signals can pass in either direction over a single transmission line simultaneously, the data communication mode is called full-duplex. An example of a full-duplex is a telephone line where the two channels are created as separate frequency bands. Cost permitting, two separate transmission lines can be established in which case the full-duplex communication is conducted over two simplex channels. This requires duplication of all the functions of a simple data communication system as shown in Figure 16.1.

## Unbalanced vs. Balanced Transmission

Implementation of the electrical transmission line can take two basic forms, unbalanced (single-ended) or balanced (differential). For unbalanced operation, a single conductor is used to carry the signal voltage, which is referenced to a signal ground. The signal ground is usually common return for all signals in the interface. Figure 16.2 shows an example of an unbalanced data transmission system with two channels and three wires. Symbol D represents driver and symbol R receiver. Unbalanced data transmission is

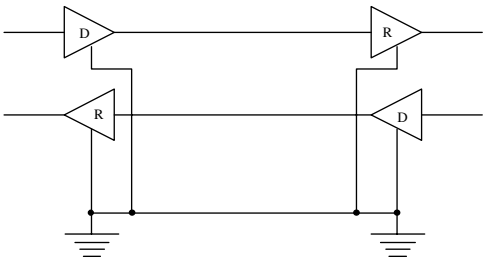



**FIGURE 16.2** Example of an unbalanced data transmission.

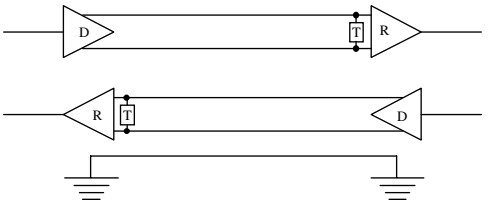



**FIGURE 16.3** Example of a balanced data transmission.

relatively inexpensive because, for multiple signal lines, only one common line is required; however, this type of interface is susceptible to induced and ground noise and is not suitable for high-speed communication over long distances. The ground noise is associated with voltage drop in a common return line, while the induced noise comes from interfering electromagnetic fields. Both types of noise can come from external sources or from neighboring transmission circuits. A remedy can be the use of coaxial cable, shielded cable, and/or the use of separate return lines for individual signals. These additional measures tend to increase the cost of the interface.

The balanced (differential) transmission mode has much better noise immunity than the unbalanced mode. Two complementary signal lines carry the data signal. The implementation often involves two single-ended drivers driving a twisted-pair transmission line. Figure 16.3 shows an example of balanced data transmission with two channels and five wires. As in Figure 16.2, symbol D represents driver and symbol R receiver. Symbol T represents termination resistor. Use of a termination resistor at the receiver end of the transmission line is critical for high-speed communications over long distances as unterminated transmission lines can cause severe distortion of signals. Both induced and ground noises appear on both conductors as common-mode signals that are rejected by the differential receiver. The differential signals carrying data are amplified while the common-mode noise signals are suppressed. As a result, the balanced data transmission lines can be used for longer distances with higher transmission rates. Both unbalanced and balanced interfaces shown in Figures 16.2 and 16.3 represent two simplex interfaces, which can form one full-duplex point-to-point (see below) communication channel.

A good source of information on individual drivers and receivers is provided in the data sheets and application notes of semiconductor manufacturers [1,2].

## Point-to-Point vs. Multi-Point

If communication takes place between two devices, we call such a communication link a point-to-point link. In mechatronic systems, it is often required for the master system to communicate with a number of subsystems. Cost permitting, a number of point-to-point data transmission lines can be implemented. In a point-to-point arrangement, the master system has a point-to-point connection to each individual subsystem, i.e., there is a separate port and communication line for each subsystem. This type of arrangement is shown in Figure 16.4. The connection can also be arranged as a multi-point connection in which all devices are connected to a single transmission line, as shown in Figure 16.5. This arrangement is a

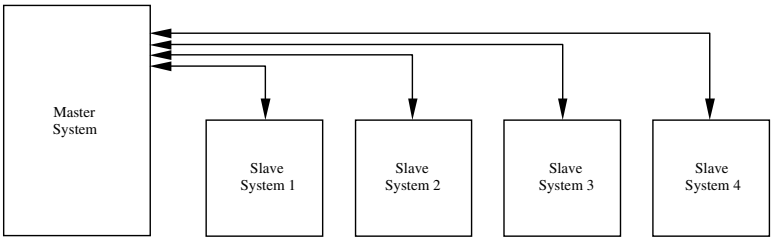



**FIGURE 16.4** A point-to-point communication system with four subsystems.

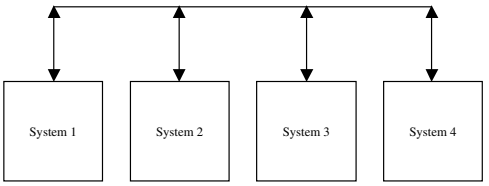



**FIGURE 16.5** A multi-point communication system.

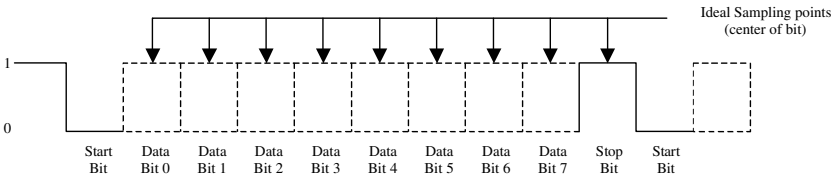



**FIGURE 16.6** Asynchronous serial data format.

data communication network arrangement where data can be transmitted from any device to any other device on the network. All devices on the network must be equipped with a receiver and a transmitter. Transmitters must have a tri-state (high output impedance) capability so that they do not provide additional load to the line. When transmitters are not transmitting, they are virtually disconnected from the transmission line. Complex communication protocol is required to manage individual transmitters on the network. The major advantage of a multi-point arrangement is usually the lower cost of the network compared to the individual communication links. The disadvantage is a more complex communication protocol (which must deal with the identity of the transmitting and receiving devices) and a more complex interface.

## Serial Asynchronous Communications

In asynchronous serial communications, the data are transmitted at irregular intervals as a bit stream. Individual characters coded as binary numbers are converted to serial data streams, which are framed with start and stop bits. Optionally, a parity bit is added to the stream. In general, a computer represents information in parallel form such as bytes and words while the majority of communications with external devices takes place serially. The task of the parallel-to-serial and serial-to-parallel conversion is performed by a special integrated circuit called a universal asynchronous receiver transmitter (UART) as described later (see Figure 16.7).

Figure 16.6 shows an example of a typical data stream for asynchronous transmission. During idle time the line is in logical state 1 (for historical reasons also called "MARK"). The start of the data stream is always indicated by the start bit, which has logical value 0 (also called "SPACE"). The start bit is followed by 5–8 data bits representing a character. The data bits are followed by an optional parity bit. The stream

is terminated by one or two stop bits with logical value 1, which can be followed by idle line or the start bit of the next character. The idle line corresponds to logical state 1. A parity bit is an extra bit inserted after the data bits and before the stop bit(s). It is set according to the parity information of the data in the stream. For example, if an even parity is used, the parity bit is set such that total number of ones in the data stream including the parity bit is even. The parity bit is used by the receiver for error checking. The task of the receiver is to detect the start of the data stream and to correctly sample individual bits in the stream. After the detection of the start bit the receiver should sample individual bits, ideally at the mid point of each bit, as shown in Figure 16.6. In the case of an ideal sampling, as shown in Figure 16.6, the receiver is said to have distortion tolerance of 50%. In practice, the receiver of a UART is sampling incoming signals using the Baud Rate Generator frequency, which is 16 times higher than the corresponding baud rate used for transmission. The uncertainty in the detection of the start bit will reduce the distortion tolerance by 6.25% (1/16) to 43.75% [3].

If, for example, the receiver clock is 1% slower than the clock of the corresponding transmitter, the sampling time of the first data bit will be delayed by 1.5% of the bit time and the sampling time of the stop bit will be delayed by 9.5%. In this case, the distortion tolerance would be further reduced to 34.25%. If the receiver clock is slower by 5%, then the receiver may detect the start bit of the next character instead of the stop bit of the current character. This results in a framing error. The above example shows the significance of the accuracy of the clock speed and the reason why the data stream must be kept short in asynchronous transmission.

Other factors affecting the error-free communications include length and type of transmission line, speed of communications, parameters of line drivers, termination of transmission line, and the level of noise in the communication system.

## The Universal Asynchronous Receiver Transmitter (UART)

The basic function of the UART is to facilitate parallel-to-serial and serial-to-parallel data conversion. The UART usually contains one transmitter and one receiver. The receiver and transmitter can operate simultaneously and independently. The UART can operate in full-duplex or half-duplex mode.

Parallel data from the host computer are converted to an asynchronous serial bit stream. The UART automatically adds a start bit, an optional parity bit, and the programmed number of stop bits, and sends the stream out through the transmitter serial data output (TxD) output pin. The parallel data are converted to a serial stream with the least significant bit shifted out first. Figure 16.7 shows a typical arrangement for UART. As can be seen, the UART uses TTL (transistor transistor logic) compatible interface. The TIA/EIA (see later) transmission line drivers and receivers are specific to a particular interface; thus, changing system interface means changing the transmission medium and the relevant drivers and receivers. The use of UART is independent of the transmission medium.

Serial data received on the receiver serial data input (RxD) pin is converted to parallel data. In the process the UART checks the start bit, parity bit (if any), and stop bit and reports any error conditions. Note that the UART is capable of generating all signals required for successful bit-serial asynchronous communications.

The UART can also report a number of error conditions, including receiver overrun, parity error, framing error, and break error. Receiver overrun error occurs when the bytes are received faster than the computer processes them. The parity error is indicated if the parity of the bit stream changed during the

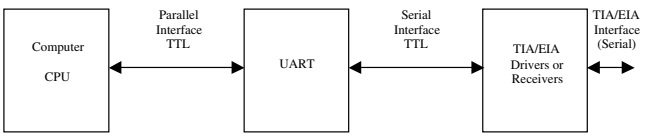



**FIGURE 16.7** Typical arrangement for the UART.

communication process. Framing error is reported if the sampled stop bit is not at logic 1 level. The break error is reported if the communication line is idle for the time equivalent to the duration of at least one character.

Older types of UART devices such as 8250, 16450 had only one byte FIFO (first in first out) buffer and thus it was easy to overrun the receiver buffer. More recent devices are equipped with larger buffers providing more efficient communications. For example, device 16550D from National Semiconductor has a 16-byte receiver buffer and a 16-byte transmitter buffer and can operate at speeds up to 1.5 Mbps. Modern UARTs can also automatically handle tasks pertaining to multi-drop systems on a network.

## 16.2 TIA/EIA Serial Interface Standards

### RS-232 Serial Interface

The RS-232 (Recommended Standard) was originally developed in 1962 by the Electronic Industries Association (EIA) as an interface between a computer and communication equipment. It is now jointly maintained by the Telecommunication Industries Association (TIA) and the EIA. The current version is designated as TIA/EIA 232-F (sixth revision) [4]. The Consulting Committee for International Telegraphs and Telephones (CCITT) issues recommendations that cover interfaces equivalent or similar to those issued by TIA/EIA.

Rapid development of computers created a demand for computer-to-computer communications over long distances. The switched public telephone network provided a readily available infrastructure for the communication task. Because computers generate digital data while the telephone network was designed for the transmission of voice signal, the digital signals from the computer had to be converted to a modulated signal which can be transmitted over the analog network. Modems (modulator/demodulator) are used to convert the digital signal into a modulated analog signal that is transmitted over the telephone line and converted back to digital signal by the modem at the other end of the telephone line. The RS-232 was designed as an interface between a computer and a modem. The formal name of the RS-232 standard is "Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange," in which the Data Terminal Equipment (DTE) represents the computer and the Data Communication Equipment (DCE) represents the modem. Figure 16.8 shows an example of the RS-232 interface in the system providing computer-to-computer communication over the switched telephone network. The computers at each end represent DTE and the modems represent DCE.

The RS-232 interface standard specifies mechanical, electrical, and functional characteristics of the DTE/DCE interface. The CCITT V.24 interface describes equivalent functional characteristics and relies on other standards for mechanical and electrical characteristics of the interface. The RS-232 standard is widely used in applications where it provides a direct point-to-point connection between two computers or computers and field elements of mechatronic systems in which case we are dealing with DTE to DTE interface. As this is a situation where a modem is not required, the cable used to connect a DTE to another DTE is called a "null modem" cable, which has internal built-in connections to fake the presence of a modem.

The mechanical characteristic is concerned with the actual physical connection of the DTE and DCE and involves specification of pin assignments and genders of the connectors. The RS-232 standard does not specify a connector type, but it is customary to use either 25-pin D-type (DB-25) connector, which

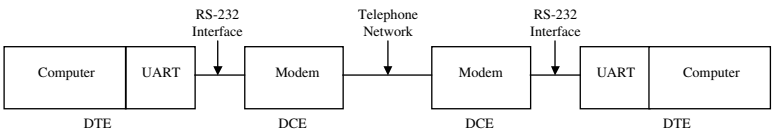



**FIGURE 16.8** Data communication over a telephone network.

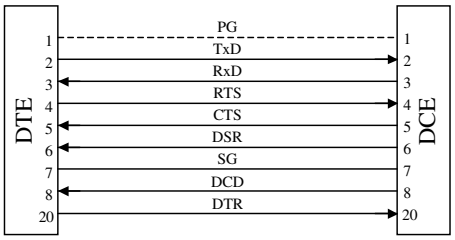



**FIGURE 16.9** Pin assignment between a DTE and DCE.

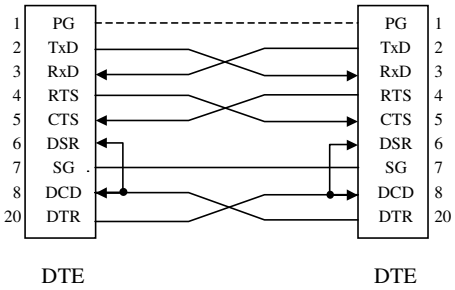



**FIGURE 16.10** Example of a null modem cable pin assignment.

can accommodate all 25 pins, listed in the standard. In practice, a smaller number of pins are used; thus, as an alternative, a 9-pin D-type connector (DB-9) is often used. Please note that the pin assignment for DB-9 connector is not specified by RS-232 and is different from DB-25 pin assignment.

Figure 16.9 shows DB-25 connector pin assignments and the interconnection of selected circuits between a DTE and a DCE. Figure 16.10, on the other hand, shows an example of the DB-25 connector pin assignments and interconnection of selected circuits between two computers, i.e., two DTEs.

## Functional Description of Selected Interchange Circuits

A full description of all signals as specified by the RS-232 standard is beyond the scope of this chapter. The reader is referred to the relevant standard [4]. We will describe the most common signals used in DTE/DCE and DTE/DTE interface. Please note that with the exception of the Protective Ground circuit and the Signal Ground circuit the circuits carry signals unidirectionally, as shown by arrows in Figure 16.9. Functional characteristics specify the functions that are performed by individual interchange circuits.

*Protective Ground (PG)*. This line ensures that the chassis of the DTE and DCE are on the same potential.

*Transmitted Data (TxD)*. Transmission line-signal originating from the DTE propagates to DCE.

*Received Data (RxD)*. Receiver line-signal originating from the DCE propagates to DTE.

*Request to Send (RTS)*. This signal is used to condition DCE for data transmission. On a half-duplex channel the signal controls the direction of data transmission of the DCE (transmit or receive). On a one-way-only channel (simplex) and on the full-duplex channels this signal controls the transmit state of the DCE (transmit or nontransmit state). A signal originating from the DTE propagates to DCE.

*Clear to Send (CTS)*. This signal indicates that the DCE is ready to receive and is the response to the asserted RTS signal. The signal is originating from the DCE and propagates to DTE.

*Data Set Ready (DSR)*. This signal indicates that the DCE is ready to operate. The signal is originating from the DCE and propagates to DTE.

*Signal Ground (SG)*. This line is a common ground return line for all other signals.
*Data Carrier Detect (DCD)*. This signal indicates that the DCE is receiving a valid modulated signal from the DCE at the other end. The signal is originating from the DCE and propagates to the DTE.
*Data Terminal Ready (DTR)*. This signal indicates that the DTE is powered up and ready to operate. This signal is originating from the DTE and propagates to DCE.

The RS-232 specifies unbalanced, unidirectional, point-to-point interface. The interconnection is done over a set of wires referred to as interchange circuits. The electrical characteristics specify voltage levels of signals, rate of change of signals, and line impedance of interchange circuits. The standard specifies Nonreturn to Zero (NRZ) coding of digital signals.

The standard requires that the drivers be designed such that for the terminator load resistance between 3 and 7 kΩ the drivers should be capable of delivering high-level voltages between +5 and +15 V and low voltages between −5 and −15 V. The electrical signals are designed to provide a 2 V margin in signaling levels. The receiver signals are defined as +3 to +15 V for high voltage and as −3 to −15 V for low voltage.

It should be noted that for the data interchange circuit the high level voltage is defined as logic 0 (SPACE), while the low level voltage is defined as logic 1 (MARK). For control signals, on the other hand, the high level voltage defines the ON state while the low level voltage defines the OFF state. The maximum rate of change of signal allowed on both data and signal lines is 30 V/$\mu$s.

The original standard has also specified the maximum length of cable as 15 m. This specification was replaced by the specification of the maximum allowed capacitive load of 2500 pF in the EIA/TIA-232-D. The maximum cable length is determined by the capacitance of the cable per unit length; thus, this parameter now defines indirectly the length of the interface cable. The RS-232 interface is rated at signaling rates in the range from 0 to 20 kbps. It should be noted that in practice a good design would allow greater distances and greater data rates than the ones specified by the standard.

## RS-422 and RS-485 Interfaces

The TIA/EIA-422-B standard "Electrical Characteristics of Balanced Voltage Digital Interface Circuits" [5] defines electrical characteristics of RS-422 interface. The RS-422 specifies a unidirectional, single driver, terminated balanced interface. The standard allows multiple receivers (up to 10) on one line. Figure 16.3 illustrates a typical point-to-point application of RS-422. As a result of improved noise immunity the RS-422 interface supports data rates up to 10 Mbps and cable length up to 1200 m, although not simultaneously. The maximum data rate of 10 Mbps is supported on a cable length up to 12 m, while a cable length of 1200 m supports data rates up to 100 kbps. Observe that the product of cable length and data rate is a limiting parameter of the interface. The transmission medium is a twisted-pair transmission line.

The TIA/EIA-485-A standard "Standard for Electrical Characteristics of Generators and Receivers for Use in Digital Multipoint Systems" [6] defines electrical characteristics of RS-485 interface. The RS-485 is a unique standard, which allows multiple nodes to communicate bidirectionally over a single twisted-pair transmission line. The RS-485 standard defines a low cost, multipoint balanced interface with electrical characteristic, supported cable types, cable length and data rates equivalent to those specified by the RS-422 standard. RS-485 parts are backward compatible and interchangeable with their equivalent RS-422 parts; however, RS-422 parts should not be used in RS-485 systems. RS-422 is usually used in point-to-point full-duplex communication systems, while RS-485 is used in multipoint half-duplex communication systems. The distinguishing feature of RS-485 drivers is their TRI-STATE capability, which allows the use of multiple drivers. The RS-485 has improved driver capability and input voltage range and supports up to 32 devices (drivers and/or receivers) on a single transmission line. Figure 16.11 shows a typical RS-485 multipoint application.

It should be noted that both RS-422 and RS-485 are electrical standards only. They do not specify mechanical or functional requirements.

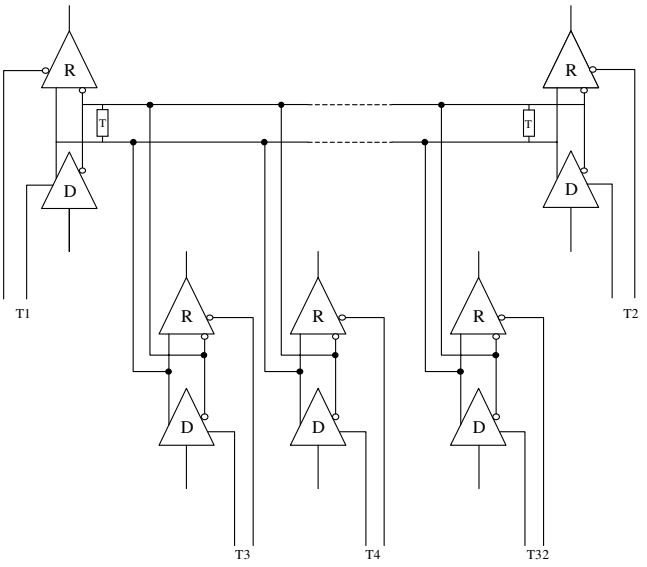



**FIGURE 16.11** Example of an RS-485 multipoint application.

As mentioned earlier, data sheets and application notes of component suppliers provide an excellent source of information [1,2]. An excellent overview of practical data communications and interfacing for instrumentation and control is provided in [7]. Detailed discussion of design aspects of serial communications and interfacing based on RS-232 and RS-485 is given in [8]. It is recommended that for the full specification the designer should consult the relevant standards [4–6]. Additional information including design recommendations can be found in [9–11]. A good introduction and background theory to data communication and computer networks can be found in [12].

## 16.3 IEEE 488—The General Purpose Interface Bus (GPIB)

### Introduction

The interface described by IEEE 488 standard, which will be referred to as GPIB in this chapter, is used to connect instruments to test and measurement systems. Examples of such instruments are digital voltmeters, storage oscilloscopes, printers, and plotters. In general, these instruments are called GPIB devices. These devices operate under the coordination of a controller. Most modern systems consist of a cluster of such devices connected to one or more computers. In such a system, one of the computers will become the controller.

Historically, the interface was developed by Hewlett–Packard in 1965. At that time, the interface was called HPIB, and a general standard did not exist. In 1975, it was formulated as IEEE 488 and was called IEEE Standard Digital Interface for Programmable Instrumentation. The standard specified the electrical, mechanical, and hardware aspects, i.e., the signals, their functioning, and purpose. Instrument manufacturers used the interface freely without adhering to a standard protocol in communicating with instruments. Instruments meant for the same purpose, yet manufactured by different manufacturers required widely varied commands. Some instruments made measurements in response to a command, while some other instruments of similar type made measurements without a command at all. Further, there were no agreed data formats between instruments sending data and instruments receiving data. This situation led to the development of an extension to the IEEE 488 standard. The new standard was published in 1987 and was

called IEEE 488.2 Standard Codes, Formats, Protocols, and Common Commands for Use with IEEE 488.1 (1987) [13], where IEEE 488.1 is the new name for the original IEEE 488 standard.

The IEEE 488.2 compliant devices must present data through data formats and codes specified in the standard. The standard also specifies a minimum set of mandatory control sequences or commands and suggests a few other optional commands. It also provides a standard status-reporting model that must be implemented by the instrument manufacturers so that determining the status of instruments will be easier for the instrument programmers.

Although not yet a standard, The Standard Commands for Programmable Instrumentation (SCPI) put together in 1990 agrees upon a standard set of commands for various instrument categories. Accordingly, all digital voltmeters manufactured by different manufacturers will respond to the same GPIB command.

## GPIB Hardware

This section describes the electrical and mechanical specifications of the GPIB interface as well as the signal description and their purpose.

All GPIB devices are connected using a special cable with each end having the male as well as the female ends of the connector. This permits piggyback connections of cables. The devices can be connected either in a chained manner (i.e., device B connected to device A, device C connected to device B, etc.) or in a star configuration (i.e., device A, B, C, etc. connected to a common node). The connection configurations are shown in Figure 16.12.

A maximum of 15 devices can be connected to the bus. The maximum separation between two devices is 4 m with an average separation of not more than 2 m. At least two-thirds of the devices connected must be powered on.

The GPIB cable consists of 24 wires. Eight of these lines are data lines, while three lines are used for handshaking. Another five lines are used for interface management and the remaining eight lines are ground lines. Among the ground lines are a cable shield line, a signal ground line, three ground return lines for the handshaking signals, and three other ground return lines for three of the interface management lines. All signals used are standard TTL signal levels with negative logic. The handshake lines and interface management lines are given in Table 16.1.

The operation of the individual lines is not important to the average user or programmer as their usage is taken care of by the controllers and the instruments that comply with IEEE 488.2 standard.

**TABLE 16.1** Handshaking and Interface Management Lines

| Handshaking Lines | | Interface Management Lines | |
|---|---|---|---|
| NRFD | Not ready for data | ATN | Attention |
| NDAC | Not data accepted | IFC | Interface clear |
| DAV | Data valid | REN | Remote enable |
| | | SRQ | Service request |
| | | EOI | End of identify |

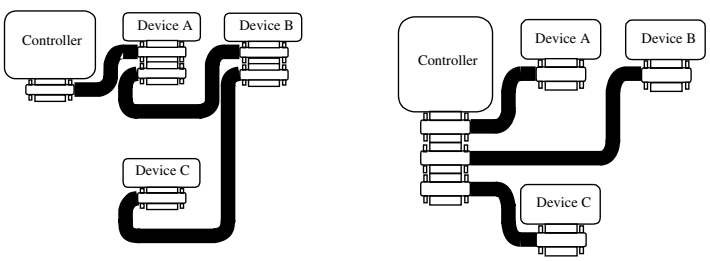



**FIGURE 16.12** Linear and star configurations of connecting GPIB devices to a controller.

## Controllers, Talkers, and Listeners

The controller carries out the general management of the bus. While there can be many controllers connected to the GPIB network, there can be only one controller-in-charge (CIC) which manages the bus at that given time. All information sent out by the controller on the data lines are called "commands" and all information sent out by other devices are termed "data." The GPIB devices that send data any time are called "talkers" and the devices that receive data are called "listeners." While there can be more than one listener operating at any given time, there can be only one talker operating at any given time. A system can have permanent talkers and permanent listeners; however, if the capability exists, a GPIB device can be a listener at one time and a talker at another time. A brief explanation of the signal lines is given below, as it would enhance our understanding of the operation of GPIB interface.

## Interface Management Lines

### Attention (ATN)

The ATN line is controlled by the CIC. When asserted, the signals on the data lines constitute a command signal and all devices must listen. When unasserted, the signals on the data lines represent data and are generally sent by a talker to one or more listeners.

### Interface Clear (IFC)

The IFC line is asserted by the CIC to reset the GPIB bus. Upon receipt of this signal, all GPIB devices on the bus will initialize themselves.

### Remote Enable (REN)

GPIB devices can be controlled either locally or remotely. The CIC asserts the REN line to bring all GPIB devices under remote programming mode. Thus, for example, the change of scale of a DVM can be carried out by a GPIB command instead of a front panel control.

### Service Request (SRQ)

Any device other than a controller can asynchronously assert the SRQ line requesting service from the controller. The controller monitors the SRQ line and polls all devices to determine the device or devices requiring service.

### End of Identity (EOI)

The EOI signal is used by a talker to indicate the end of the data message of the talker. It indicates to the listener(s) the end of the receiving data record.

## Handshake Lines

In general, a data transfer with complete handshake gets through three stages: request or preparedness, data transfer, and acknowledgment. On some systems, where the stability of data on the data bus is questionable, a data valid signal may also be provided. On the GPIB bus, when a talker has to send data to a listener, the controller must address a device and instruct it to be the talker and then address one or more other devices and instruct them to be listeners. See later for "Addressing of GPIB Devices."

### Not Ready for Data (NRFD)

The NRFD line is controlled by the controller when sending commands or by the talker when sending data. A device that has been instructed to be a listener will unassert NRFD to indicate to the talker that it is ready to receive data. Of all the listeners, the slowest device will be the last to unassert NRFD and thus control the speed of data transfer.

### Data Valid (DAV)

When all listeners have indicated their readiness to receive data by unasserting NRFD, the talker (or the controller when sending commands) will assert a DAV signal to indicte to all listeners that the data on the data lines DIO1-DIO8 are stable and may be read by the listeners. In response to a DAV signal, the listeners

may assert NRFD to halt any further data transmission by talkers until the data already transmitted has been received.

### Not Data Accepted (NDAC)

The NDAC line driven by all listeners is the acknowledgment signal. When data has been received by all listeners, the NDAC line will be unasserted. The talker can then remove the data and unassert the DAV signal.

## Data Lines DIO1-DIO8 (8 lines)

The data lines are controlled by the controller when issuing commands or by the talker. As soon as the controller instructs a particular device to be a talker, that device will place data on the lines DIO1-DIO8 and will wait for at least T1 seconds.

## Addressing of GPIB Devices

All GPIB devices connected to a GPIB bus must have a unique GPIB address. A device can have a primary address as well as a secondary address. Most devices use a primary address only. The addresses are in the range 0–30 decimal. In general, the addresses on the instruments as well as the controller are set using switches. The controller instructs a device with a particular address to be a talker or a listener by sending a bit pattern on the data bus. The bit pattern is formed according to Table 16.2. The data bits are numbered from D7 to D0. The value of each bit is listed straight underneath. The letter A signifies 0 or 1. The five bits D4–D0 will form a bit pattern representing the address of the device. The letter X signifies a "don't care" bit, which is not used. TA will be set to 1 if the controller is instructing the device to be a talker. LA will be set to 1 if the controller is instructing the device to be a listener. For example, if a particular device has the address 15 (decimal) and if the controller is instructing that device to be the talker, then the controller must send the following bit pattern over the data lines DIO8-DIO1.

Device with address 15 be the talker 0 1 0 0 1 1 1 1 = 4F (hex)

Similarly,

Device with address 0 be the listener 0 0 1 0 0 0 0 0 = 20 (hex)

"Untalk" the current talker 0 1 0 1 1 1 1 1 = 5F (hex)

"Unlisten" all listeners 0 0 1 1 1 1 1 1 = 3F (hex)

Note that Untalk and Unlisten commands look very similar to Talk and Listen commands; however, the address 31 (decimal) does not exist. Therefore, the address 31 is used to affect Untalk and Unlisten.

For the controllers, IEEE 488.2 Standard provides the Required and Optional Control Sequences. All controllers that comply with IEEE 488.2 must support all mandatory commands. The standard also provides the "Controller Protocols." Protocols are formed by combining a set of control sequences. For example, FINDLSTN command will issue a set of control sequences to determine the existing listeners.

For the instruments, IEEE 488.2 specifies a set of mandatory commands and queries. For example, when the command "*RST" is received by IEEE 488.2 compliant instruments, they all must carry out an instrument reset. Similarly, upon receipt of "*STB?" command the instrument will send the status byte to the controller.

**TABLE 16.2** Talker/Listener Addressing Commands

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| X | TA | LA | A | A | A | A | A |

All mandatory common commands and queries, all required and optional control sequences as well as the controller protocols can be found in [14,15].

## References

1. Goldie, J., Summary of well known interface standards, Application note AN-216, National Semiconductor, 1998, www.national.com.
2. Goldie, J., Comparing EIA-485 and EIA-422-A line drivers and receivers in multipoint applications, Application note AN-759, National Semiconductor, 1998.
3. McNamara, J.E., *Technical Aspects of Data Communication,* 3rd edition, Digital Press, 1988.
4. TIA/EIA-232-F, Interface between data terminal equipment and data communication equipment employing serial binary data interchange, TIA, EIA, 1997.
5. TIA/EIA-422-B, Electrical characteristics of balanced voltage digital interface circuits, TIA, EIA, 1995.
6. TIA/EIA-485-A, Standard for electrical characteristics of generators and receivers for use in digital multipoint systems, TEI, EIA, 1998.
7. Mackay, S.G., et al., *Data Communications for Instrumentation and Control,* IDC Techbooks, 2000.
8. Axelson, J., *Serial Port Complete,* Lakeview Research, Madison, 1998.
9. Goldie, J., Ten ways to bulletproof RS-485 interfaces, Application note AN-1057, National Semiconductor, 1996.
10. RS-422 and RS-485 Application Note, B&B Electronics Manufacturing Co., 1997, www.bb-elec.com.
11. DALLAS SEMICONDUCTOR, Application Note 83, Fundamentals of RS-232 Serial Communications, 1998.
12. Stallings, W., *Data and Computer Communications,* 6th ed., Prentice-Hall, Upper Saddle River, NJ, 2000.
13. ANSI/IEEE 488.1-1987 IEEE standard digital interface for programmable instrumentation institution of electrical and electronic engineers, New York, 1987.
14. ANSI/IEEE 488.2-1987 IEEE standard codes, formats, protocols and common commands, institution of electrical and electronic engineers, New York, 1987.
15. ANSI/IEEE 488.2-1992 IEEE standard codes, formats, protocols and common commands, and standard commands for programmable instruments. Institution of Electrical and Electronic Engineers, New York, 1992.

# 17

# Communications and Computer Networks




Mohammad Ilyas
*Florida Atlantic University*


The field of communications and computer networks deals with efficient and reliable transfer of information from one point to another. The need to exchange information is not new but the techniques employed to achieve information exchange have been steadily improving. During the past few decades, these techniques have experienced an unprecedented and innovative growth. Several factors have been and continue to be responsible for this growth. The Internet is the most visible product of this growth and it has impacted the life of each and every one of us. This chapter describes salient features and operational details of communications and computer networks.

The contents of this chapter are organized in several sections. Section 17.1 describes a brief history of the field of communications. Section 17.2 deals with the introduction of communication and computer networks. Section 17.3 describes operational details of computer networks. Section 17.4 discusses resource allocation mechanisms. Section 17.5 briefly describes the challenges and issues in communication and computer networks that are still to be overcome. Finally, Section 17.6 summarizes the article.

## 17.1 A Brief History

Exchange of information (communications) between two or more entities has been a necessity since the existence of human life. It started with some form and shape of human voice that one entity can create and other(s) can listen to and interpret. Over a period of several centuries, these voices evolved into languages. As the population of the world grew, more and more languages were born. For a long time, languages were used for face-to-face communications. If there were ever a need to convey some information (a message) over a distance, someone would be briefed and sent to deliver the message to a distant site. Gradually, additional methods were developed to represent and exchange the information. These methods included symbols, shapes, and eventually alphabets. This development facilitated information recording and use of nonvocal means for exchanging information. Hence, preservation, dissemination, sharing, and communication of knowledge became easier.

Until about 150 years ago, all communication was via wireless means and included smoke signals, beating of drums, and use of reflective surfaces for reflecting light signals (optical wireless). Efficiency of

these techniques was heavily influenced by environmental conditions. For instance, smoke signals were not very effective in windy conditions. In any case, as we will note later, some of the techniques that were in use centuries ago for conveying information over a distance were similar to the techniques that we currently use. The only difference is that the implementation of those techniques is exceedingly more sophisticated now than it was centuries ago.

As the technological progress continued and electronic devices started appearing on the surface, the field of communication also started making use of the innovative technologies. Alphabets were translated into their electronic representations so that information could be electronically transmitted. Morse code was developed for telegraphic exchange of information. Further developments led to the use of the telephone. It is important to note that in earlier days of technological masterpieces, users would go to a common site where one could send a telegraphic message over a distance or could have a telephonic conversation with a person at a remote location. This was a classic example of resource sharing. Of course, human help was needed to establish a connection with remote sites.

As the benefits of the advances in communication technologies were being harvested, electronic computers were also emerging and making the news. Earlier computers were not only expensive and less reliable, they were also huge in size. For instance, the computers that used vacuum tubes were of the size of a large room and used roughly about 10,000 vacuum tubes. These computers would stop working if a vacuum tube burned out, and the tube would need to be replaced by using a ladder. On average, those computers would function for a few minutes before another vacuum tube's replacement was necessary. A few minutes of computer time was not enough to execute a large computer program. With the advent of transistors, computers not only became smaller in size and less expensive, but also more reliable. These aspects of computers resulted in their widespread applications. With the development of personal computers, there is hardly any side of our lives that has not been impacted by the use of computers. The field of communications is no exception and the use of computers has escalated our communication capabilities to new heights.

## 17.2 Introduction

Communication of information from one point to another in an efficient and reliable manner has always been a necessity. A typical communication system consists of the following components as shown in Figure 17.1:

- Source that generates or has the information to be transported
- Transmitter that prepares the information for transportation
- Transmission medium that carries the information from one end to the other
- Receiver that receives the information and prepares it for delivering to the receiver
- Destination that takes the information from receiver and utilizes it as necessary

The information can be generated in analog or digital form. Analog information is represented as a continuous signal that varies smoothly in time. As one speaks in a microphone, an analog voice signal is generated. Digital information is represented by a signal that stays at some fixed level for some duration of time followed by a change to another fixed level. A computer works with digital information that has two levels (binary digital signals). Figure 17.2 shows an example of analog and digital signals. Transmission of

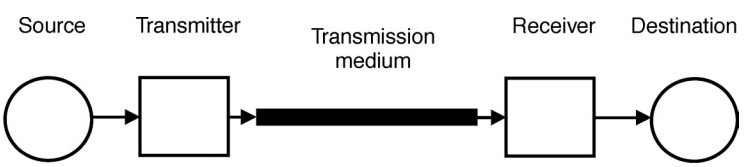



**FIGURE 17.1** A typical communication system.

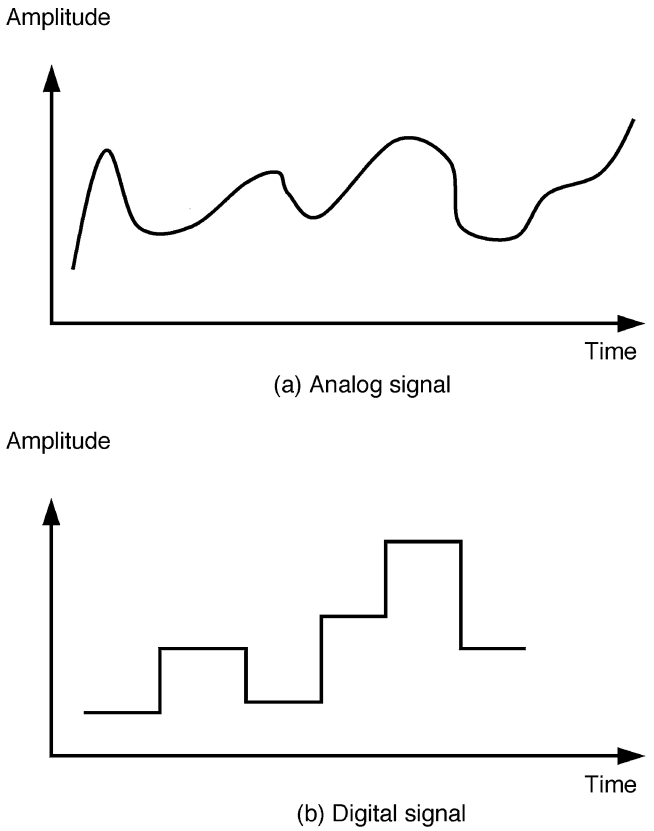



**FIGURE 17.2** Typical analog and digital signals.

information can also be in analog or digital form. Therefore, we have the following four possibilities in a communication system [21]:

- Analog information transmitted as an analog signal
- Analog information transmitted as a digital signal
- Digital information transmitted as an analog signal
- Digital information transmitted as a digital signal

There may not be a choice regarding the form (analog or digital) of information being generated by a device. For instance, a voice signal as one speaks, a video signal as generated by a camera, a speed signal generated by a moving vehicle, and an altitude signal generated by the equipment in a plane will always be analog in nature; however, there is a choice regarding the form (analog or digital) of information being transmitted over a transmission medium. Transmitted information could be analog or digital in nature and information can be easily converted from one form to another.

Each of these possibilities has its pros and cons. When a signal carrying information is transmitted, it looses its energy and strength and gathers some interference (noise) as it propagates away from the transmitter. If the energy of the signal is not boosted at some intermediate point, it may attenuate beyond recognition before it reaches its intended destination. That will certainly be a wasted effort. In order to boost energy and strength of a signal, it must be amplified (in case of analog signals) and rebuilt (in case of digital signals). When an analog signal is amplified, the noise also becomes amplified and that certainly lowers expectations about receiving the signal at its destination in its original (or close to it) form. On the other hand, digital signals can be processed and reconstructed at any intermediate point and, therefore, the noise can essentially be filtered out. Moreover, transmission of information in digital form has many

other advantages including processing of information for error detection and correction, applying encryption and decryption techniques to sensitive information, and many more. Thus, digital information transmission technology has become the dominant technology in the field of communications [9,18].

As indicated earlier, communication technology has experienced phenomenal growth over the past several decades. The following two factors have always played a critical role in shaping the future of communications [20]:

- Severity of user needs to exchange information
- State of the technology related to communications

Historically, inventions have always been triggered by the severity of needs. It has been very true for the field of communications as well. In addition, there is always an urge and curiosity to make things happen faster. When electricity was discovered and people (scattered around the globe) wanted to exchange information over longer distances and in less time, the telegraph was invented. Morse code was developed with shorter sequences (of dots and dashes) for more frequent alphabets. That resulted in transmission of messages in a shorter duration of time. The presence of electricity and the capability of wires to carry information over longer distances led to the development of devices that converted human voice into electrical signal, and thus led to the development of telephone systems. Behind this invention was also a need/desire to establish full-duplex (two-way simultaneous) communication in human voice. As use of the telephone became widespread, there was a need for a telephone user to be connected to any other user, and that led to the development of switching offices. In the early days, the switching offices were operated manually. As the state of the technology improved, the manual switching offices were replaced by automatic switching offices. Each telephone user was assigned a telephone number for identification purposes and a user able to dial the number for the purpose of establishing a connection with the called party. As the computer technology improved and the computers became easier to afford and smaller in size, they found countless uses including their use in communications. The computers not only replaced the automatic (electromechanical) switching offices, they were also employed in many other aspects of communication systems. Examples include conversion of information from analog to digital and vice versa, processing of information for error detection and/or correction, compression of information, and encryption/decryption of information.

As computers became more powerful, there were many other applications that surfaced. The most visible application was the amount of information users started sharing among themselves. The volume of information being exchanged among users has been growing exponentially over the last three decades. As users needed to exchange such a mammoth amount of information, new techniques were invented to facilitate the process. There was not only a need for users to exchange information with others in an asynchronous fashion, there was also a need for computers to exchange information among themselves. The information being exchanged in this fashion has different characteristics than the information being exchanged through the telephone systems. This need led to the interconnection of computers with each other and that is what is called computer networks.

## 17.3 Computer Networks

A computer network is an interconnection of computers. The interconnection forms a facility that provides reliable and efficient means of communication among users and other devices. User communication in computer networks is assisted by computers, and the facility also provides communication among computers. Computer networks are also referred to as computer communication networks. Interconnection among computers may be via wired or wireless transmission medium [5,6,10,13,18].

There are two broad categories of computer networks:

- Wide area networks
- Local/metropolitan area networks

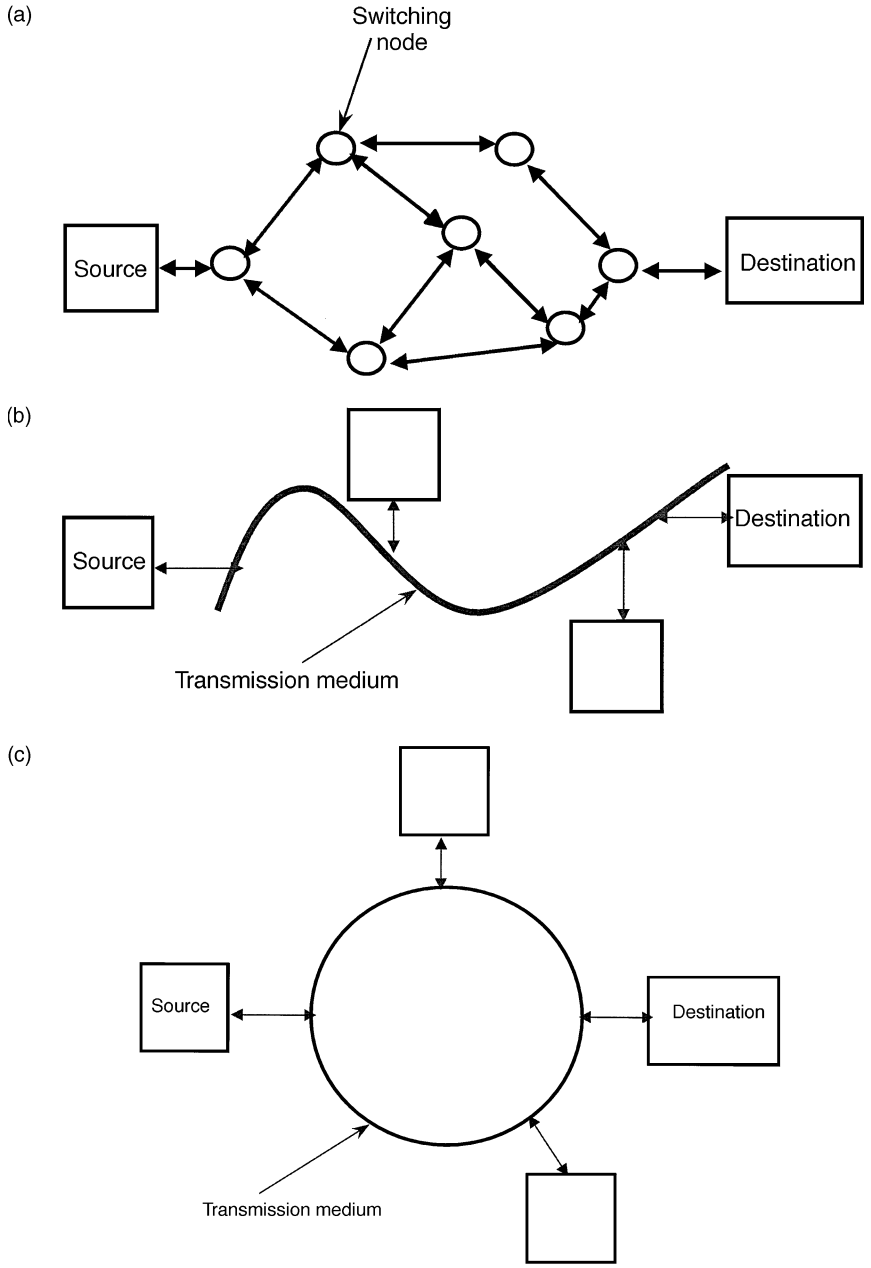



**FIGURE 17.3** (a) A typical wide area computer communication network. (b) A typical local/metropolitan area communication bus network. (c) A typical local/metropolitan area communication ring network.

Wide area computer networks, as the name suggests, span a wider geographical area and essentially have a global scope. On the other hand, local/metropolitan area networks span a limited distance. Local area networks are generally confined to an industrial building or an academic institution. Metropolitan area networks also have limited geographical scope but it is relatively larger than that of the local area networks [19]. Typical wide and local/metropolitan area networks are shown in Figure 17.3.

Once a user is connected to a computer network, that user can communicate with any other user also connected to the network at some point. It is not required for a user to be connected directly to another

user in order to communicate. In fact, in wide area networks, two communicating users will rarely be directly connected with each other. This implies that the users will be sharing the transmission links for exchanging their information. This is one of the most important aspects of computer networks. Sharing of resources improves utilization of the resources and is, of course, cost-effective as well. In addition to sharing the transmission links, the users will also share the processing power of the computers at the switching nodes, buffering capacity to store the information at the switching nodes, and any other resources that are connected to the computer network. A user who is connected to a computer network at any switching node will have immediate access to all the resources (databases, research articles, surveys, and much more) that are connected to the network as well. Of course, access to specific information may be restricted and a user may require appropriate authorization to access the information.

The information from one user to another may need to pass through several switching nodes and transmission links before reaching its destination. This implies that a user may have many options available to select one out of many sequences of transmission links and switching nodes to exchange information. That adds to the reliability of the information exchange process. If one path is not available, not feasible, or not functional, some other path may be used. In addition, for better and effective sharing of resources among several users, it is not appropriate to let any user exchange a large quantity of information at a time; however, it is not uncommon that some users may have a large quantity of information to exchange. In that case, the information is broken into smaller units known as packets of information. Each packet is sent toward its destination as a separate entity and then all packets are assembled together at the destination side to re-create the original piece of information [2].

Due to the resource sharing environment, users may not be able to exchange their information at any time they wish because the resources (switching nodes, transmission links) may be busy serving other users. In that case, some users may have to wait for some time before they begin their communication. Designers of computer networks should design the network so that the total delay (including wait time) is as brief as possible and that the total amount of information successfully exchanged (throughput) is as large as possible.

Many aspects must be addressed for enabling networks to transport users' information from one point to another. The major aspects are:

- Addressing mechanism to identify users
- Addressing mechanism for information packets to identify their source and destination
- Establishing a connection between sender and receiver and maintaining it
- Choosing a path or a route (sequence of switching nodes and transmission links) to carry the information from a sender to a receiver
- Implementing a selected route or path
- Checking information packets for errors and recovering from errors
- Encryption and decryption of information
- Controlling the flow of information so that shared resources are not over-taxed
- Informing the sender that the information has been successfully delivered to the intended destination (acknowledgment)
- Billing for the use of resources
- Ensuring that different computers running different applications and operating systems can exchange information
- Preparing information appropriately for transmission over a given transmission medium

This is not an exhaustive list of items that need to be addressed in computer networks. In any case, all such issues are addressed by very systematic and detailed procedures. The procedures are called communication protocols. The protocols are implemented at the switching nodes by a combination of hardware and software. It is not advisable to implement all these features in one module of hardware or software because that will become very difficult to manage. It is a standard practice that these features be divided into different

smaller modules and then these modules can be interfaced together to collectively provide implementation of these features. International Standards Organization (ISO) has suggested dividing these features into seven distinct modules called layers. The proposed model is referred to as Open System Interconnection (OSI) reference model. The seven layers proposed in the OSI reference model are [2]:

- Application layer
- Presentation layer
- Session layer
- Transport layer
- Network layer
- Data link layer
- Physical layer

The physical layer deals with the transmission of information on the transmission medium. The data link layer handles the information on a single link. The network layer deals with the path or route of information from the switching node where the source is connected to the switching node where the receiver is connected. It also monitors end-to-end information flow. The remaining four layers reside with the user equipment. The transport layer deals with the information exchange from the source to the sender. The session layer handles the establishment of a session between the source and the receiver and maintains it. The presentation layer deals with the form in which information is presented to the lower layer. Encryption/decryption of information can also be performed at this layer. The application layer deals with the application that generates the information at the source side and what happens to it when it is delivered at the receiver side.

As the information begins from the application layer at the sender side, it is processed at every layer according to the specific protocols implemented at that layer. Each layer processes the information and appends a header and/or a trailer with the information before passing it on to the next layer. The headers and trailers appended by various layers contribute to the overhead and are necessary for transportation of the information. Finally, at the physical layer, the bits of information packets are converted to an appropriate signal and transmitted over the transmission medium. At the destination side, the physical layer receives the information packets from the transmission medium and prepares them for passing these to the next higher layer. As a packet is processed by the protocol layers at the destination side, its headers and trailers are stripped off before it is passed to the next layer. By the time information reaches the application layer, it should be in the same form as it was transmitted by the source.

Once a user is ready to send information to another user, he or she has two options. He or she can establish a communication with the destination prior to exchanging information or he can just give the information to the network node and let the network deliver the information to its destination. If communication is established prior to exchanging the information, the process is referred to as connection-oriented service and is implemented by using virtual circuit connections. On the other hand, if no communication is established prior to sending the information, the process is called connectionless service. This is implemented by using a datagram environment. In connection-oriented (virtual circuit) service, all packets between two users travel over the same path through a computer network and, hence, arrive at their destination in the same order as they were sent by the source. In connectionless service, however, each packet finds its own path through the network while traveling towards its destination. Each packet will therefore experience a different delay and the packets may arrive at their destination out of sequence. In that case, the destination will be required to put all the packets in proper sequence before assembling them [2,10,13].

As in all resource sharing systems, allocation of resources in computer networks requires careful attention. The main idea is that the resources should be shared among users of a computer network as fairly as possible. At the same, it is desired to maintain the network performance as close to its optimal level as possible. The fairness definition, however, varies from one individual to another and depends upon how one is associated with a computer network. Although fairness of resource sharing is being evaluated, two performance parameters—delay and throughput—for computer networks are considered. The delay

is the duration of time from the moment information is submitted by a user for transmission to the moment it is successfully delivered to its destination. The throughput is the amount of information successfully delivered to its intended destination per unit time. Due to the resource sharing environment in computer networks, these two performance parameters are contradictory. It is desired to have the delay as small as possible and the throughput as large as possible. For increasing throughput, a computer network must handle increased information traffic, but the increased level of information traffic also causes higher buffer occupancy at the switching nodes and, hence, more waiting time for information packets. This results in an increase in delay. On the other hand, if information traffic is reduced to reduce the delay, that will adversely affect the throughput. A reasonable compromise between throughput and delay is necessary for the satisfactory operation of a computer network [10,11].

## Wide Area Computer Networks

A wide area network consists of switching nodes and transmission links as shown in Figure 17.3(a). Layout of switching nodes and transmission links is based on the traffic patterns and expected volume of traffic flow from one site to another site. Switching nodes provide the users access to a computer network and implement communication protocols. When a user is ready to transmit his or her information, the switching node, to which the user is connected, will establish a connection if a connection-oriented service has been opted. Otherwise, the information will be transmitted in a connectionless environment. In either case, switching nodes play a key role in determining the path of the information flow according to some well-established routing criteria. The criteria include performance (delay and throughput) objectives among other factors based on user needs. For keeping the network traffic within a reasonable range, some traffic flow control mechanisms are necessary. In late 1960s and early 1970s, when data rates of transmission media used in computer networks were low (a few thousand bits per second), these mechanisms were fairly simple. A common method used for controlling traffic over a transmission link or a path was an understanding that the sender would continue sending information until the receiver sent a request to stop. The information flow would resume as soon as the receiver sent another request to resume transmission. Basically the receiver side had the final say in controlling the flow of information over a link or a path. As the data rates of transmission media started increasing, this method was not deemed efficient. To control the flow of information in relatively faster transmission media, a sliding window scheme was used. According to this scheme, the sender will continuously send information packets but no more than a certain limit. Once the limit is reached, the sender will stop sending the information packets and will wait for the acknowledgment that the packets have been transmitted. As soon as an acknowledgment is received, the sender may send another packet. This method ensures that there are no more than a certain specific number of packets in transit from sender to receiver at any given time. Again, the receiver has control over the amount of information that the sender can transmit. These techniques for controlling the information traffic are referred to as reactive- or feedback-based techniques because the decision to transmit or not to transmit is based on the current traffic conditions.

Reactive techniques are acceptable in low to moderate data rates of transmission media. As the data rates increase from kilobits per second to megabits and gigabits per second, the situation changes. Over the past several years, there has been a manifold increase in data rates. Optical fibers provide enormously high data rates. Size of the computer networks has also experienced tremendous increase. The amount of traffic flowing through these networks has been increasing exponentially. Given that, the traffic control techniques used in earlier networks are not quite effective anymore [11,12,22]. One more factor that has added to the complexity of the situation is that users are now exchanging different types of information through the same network. Consider the example of the Internet. The geographical scope of the Internet is essentially global. Extensive use of optical fiber as transmission media provides very high data rates for exchanging information. In addition, users are using the Internet for exchanging any type of information they come across, including voice, video, and data. All these factors have essentially necessitated the use of a modified approach for traffic management in computer networks. The main factor leading to this change is that the information packets are moving so fast through the computer networks that any feedback-based (or reactive)

control will be too slow to be of any use. Therefore, some preventive mechanisms have been developed to maintain the information traffic inside a computer network to a comfortable level. Such techniques are implemented at the sender side by ensuring that only as much information traffic is allowed to enter the network as can be comfortably handled by the networks [1,20,22]. Based on the users' needs and state of the technology, providing faster communications for different types of services (voice, video, data, and others) in the same computer network in an integrated and unified manner has become a necessity. These computer networks are referred to as broadband integrated services digital networks (BISDNs). BISDNs provide end-to-end digital connectivity and users can access any type of communication service from a single point of access. Asynchronous transfer mode (ATM) is expected to be used as a transfer mechanism in BISDNs. ATM is essentially a fast packet switching technique where information is transmitted in the form of small fixed-size packets called cells. Each cell is 53 bytes long and includes a header of 5 bytes. The information is primarily transported using a connection-oriented (virtual circuit) environment [3,4,8,12,17].

Another aspect of wide area networks is the processing speed of switching nodes. As the data rates of transmission media increase, it is essential to have faster processing capability at the switching nodes. Otherwise, switching nodes become bottlenecks and faster transmission media cannot be fully utilized. When transmission media consist of optical fibers, the incoming information at a switching node is converted from optical form to electronic form so that it may be processed and appropriately switched to an outgoing link. Before it is transmitted, the information is again converted from electronic form to optical form. This slows down the information transfer process and increases the delay. To remedy this situation, research is being conducted to develop large optical switches to be used as switching nodes. Optical switches will not require conversion of information from optical to electronic and vice versa at the switching nodes; however, these switches must also possess the capability of optical processing of information. When reasonably sized optical switches become available, use of optical fiber as transmission media together with optical switches will lead to all-optical computer and communication networks. Information packets will not need to be stored for processing at the switching nodes and that will certainly improve the delay performance. In addition, wavelength division multiplexing techniques are rendering use of optical transmission media to its fullest capacity [14].

## Local and Metropolitan Area Networks

A local area network has a limited geographical scope (no more than a few kilometers) and is generally limited to a building or an organization. It uses a single transmission medium and all users are connected to the same medium at various points. The transmission medium may be open-ended (bus) as shown in Figure 17.3(b) or it may be in the form of a loop (ring) as shown in Figure 17.3(c). Metropolitan area networks also have a single transmission medium that is shared by all the users connected to the network, but the medium spans a relatively larger geographical area, up to 150 km. They also use a transmission medium with relatively higher data rates. Local and metropolitan area networks also use a layered implementation of communication protocols as needed in wide area networks; however, these protocols are relatively simpler because of simple topology, no switching nodes, and limited distance between the senders and the receivers. All users share the same transmission medium to exchange their information. Obviously, if two or more users transmit their information at the same time, the information from different users will interfere with each other and will cause a collision. In such cases, the information of all users involved in a collision will be destroyed and will need to be retransmitted. Therefore, there must be some well-defined procedures so that all users may share the same transmission medium in a civilized manner and have successful exchange of information. These procedures are called medium access control (MAC) protocols.

There are two broad categories of MAC protocols:

- Controlled access protocols
- Contention-based access protocols

In controlled access MAC protocols, users take turns transmitting their information and only one user is allowed to transmit information at a time. When one user has finished his or her transmission, the next user begins transmission. The control could be centralized or distributed. No information collisions occur and, hence, no information is lost due to two or more users transmitting information at the same time. Examples of controlled access MAC protocols include token-passing bus and token-passing ring local area networks. In both of these examples, a token (a small control packet) circulates among the stations. A station that has the token is allowed to transmit information, and other stations wait until they receive the token [19].

In contention-based MAC protocols, users do not take turns transmitting their information. A user makes his or her own decision to transmit and also faces a risk of becoming involved in a collision with another station that also decides to transmit at about the same time. If no collision occurs, the information may be successfully delivered to its destination. On the other hand, if a collision occurs, the information from all users involved in a collision will need to be retransmitted. An example of contention- based MAC protocols is carrier sense multiple access with collision detection (CSMA/CD), which is used in Ethernet. In CSMA/CD, a user senses the shared transmission medium prior to transmitting its information. If the medium is sensed as busy (someone is already transmitting the information), the user will refrain from transmitting the information; however, if the medium is sensed as free, the user transmits the information. Intuitively, this MAC protocol should be able to avoid collisions, but collisions still do take place. The reason is that transmissions travel along the transmission medium at a finite speed. If one user senses the medium at one point and finds it free, it does not mean that another user located at another point of the medium has not already begun its transmission. This is referred to as the effect of the finite propagation delay of electromagnetic signal along the transmission medium. This is the single most important parameter that causes deterioration of performance in contention-based local area networks [11,19].

Design of local area networks has also been significantly impacted by the availability of transmission media with higher data rates. As the data rate of a transmission medium increases, the effects of propagation delay become even more visible. In higher speed local area networks such as Gigabit Ethernet, and 100-BASE-FX, the medium access protocols are designed to reduce the effects of propagation delay. If special attention is not given to the effects of propagation delay, the performance of high-speed local area networks becomes very poor [15,19].

Metropolitan area networks essentially deal with the same issues as local area networks. These networks are generally used as backbones for interconnecting different local area networks. These are high-speed networks and span a relatively larger geographical area. MAC protocols for sharing the same trans-mission media are based on controlled access. The two most common examples of metropolitan area networks are fiber distributed data interface (FDDI) and distributed queue dual bus (DQDB). In FDDI, the transmission medium is in the form of two rings, whereas DQDB uses two buses. FDDI rings carry information in one but opposite directions and this arrangement improves reliability of communication. In DQDB, two buses also carry information in one but opposite directions. The MAC protocol for FDDI is based on token passing and supports voice and data communication among its users. DQDB uses a reservation-based access mechanism and also supports voice and data communication among its users [19].

## Wireless and Mobile Communication Networks

Communication without being physically tied-up to wires has always been of interest and mobile and wireless communication networks promise that. The last few years have witnessed unprecedented growth in wireless communication networks. Significant advancements have been made in the technologies that support wireless communication environment and there is much more to come in the future. The devices used for wireless communication require certain features that wired communication devices may not necessarily need. These features include low power consumption, light weight, and worldwide communication ability.

In wireless and mobile communication networks, the access to a communication network is wireless so that the end users remain free to move. The rest of the communication path could be wired, wireless,

or combination of the two. In general, a mobile user, while communicating, has a wireless connection with a fixed communication facility and rest of the communication path remains wired. The range of wireless communication is always limited and therefore the range of user mobility is also limited. To overcome this limitation, the cellular communication environment has been devised. In a cellular communication environment, a geographical region is divided into smaller regions called cells, thus the name cellular. Each cell has a fixed communication device that serves all mobile devices within that cell. However, as a mobile device, while in active communication, moves out of one cell and into another cell, service of that connection is transferred from one cell to another. This is called the handoff process [7,16].

The cellular arrangement has many attractive features. As the cell size is small, the mobile devices do not need very high transmitting power to communicate. This leads to smaller devices that consume less power. In addition, it is well known that the frequency spectrum that can be used for wireless communication is limited and can therefore support only a small number of wireless communication connections at a time. Dividing communication regions into cells allows the use of the same frequency in different cells as long as they are sufficiently far apart to avoid interference. This increases the number of mobile devices that can be supported. Advances in digital signal processing algorithms and faster electronics have led to very powerful, smaller, elegant, and versatile mobile communication devices. These devices have tremendous mobile communication abilities including wireless Internet access, wireless e-mail and news items, and wireless video (though limited) communication on handheld devices. Wireless telephones are already available and operate in different communication environments across the continents. The day is not far when a single communication number will be assigned to every newborn and will stay with that person irrespective of his/her location.

Another field that is emerging rapidly is the field of ad hoc wireless communication networks. These networks are of a temporary nature and are established for a certain need and for a certain duration. There is no elaborate setup needed to establish these networks. As a few mobile communication devices come in one another's proximity, they can establish a communication network among themselves. Typical situations where ad hoc wireless networks can be used are in the classroom environment, corporate meetings, conferences, disaster recovery situations, etc. Once the need for networking is satisfied, the ad hoc networking setup disappears.

## 17.4 Resource Allocation Techniques

As discussed earlier, computer networks are resource sharing systems. Users share the common resources as transmission media, processing power and buffering capacity at the switching nodes, and other resources that are part of the networks. A key to successful operation of computer networks is a fair and efficient allocation of resources among its users. Historically, there have been two approaches to allocation of resources to users in computer networks:

- Static allocation of resources
- Dynamic allocation of resources

Static allocation of resources means that a desired quantity of resources is allocated to each user who may use it whenever he or she needs to. If the user does not use his/her allocated resources, no one else can. On the other hand, dynamic allocation of resources means that a desired quantity of resources is allocated to users on the basis of their demands and for the duration of their need. Once the need is satisfied, the allocation is retrieved. In that case, someone else can use these resources if needed. Static allocation results in wastage of resources, but does not incur the overhead associated with dynamic allocation. Which technique should be used in a given situation is subject to the concept of supply and demand. If resources are abundant and demand is not too high, it may be better to have static allocation of resources; however, when the resources are scarce and demand is high, dynamic allocation is almost a necessity to avoid the wastage of resources.

Historically, communication and computer networks have dealt with both situations. Earlier communication environments used dynamic allocation of resources when users walked to a public call office to

make a telephone call or send a telegraphic message. After a few years, static allocation of res-ources was adopted, when users were allocated their own dedicated communication channels and these were not shared among others. In the late 1960s, the era of computer networks dawned with dynamic allocation of resources and all communication and computer networks have continued with this tradition to date. With the advent of optical fiber, it was felt that the transmission resources are abundant and can satisfy any demand at any time. Many researchers and manufacturers were in favor of going back to the static allocation of resources, but a decision to continue with dynamic resource allocation was made and that is here to stay for many years to come [10].

## 17.5 Challenges and Issues

Many challenges and issues are related to communications and computer networks that are still to be overcome. Only the most important ones will be described in this section.

High data rates provided by optical fibers and high-speed processing available at the switching nodes has resulted in lower delay for transferring information from one point to another. However, the propagation delay (the time for a signal to propagate from one end to another) has essentially remained unchanged. This delay depends only on the distance and not on the data rate or the type of transmission medium. This issue is referred to as latency vs. delay issue [11]. In this situation, traditional feedback-based reactive traffic management techniques become ineffective. New preventive techniques for effective traffic management and control are essential for achieving the full potential of these communication and computer networks [22].

Integration of different services in the same networks has also posed new challenges. Each type of service has its own requirements for achieving a desired level of quality of service (QoS). Within the networks any attempt to satisfy QoS for a particular service will jeopardize the QoS requirements for other services. Therefore, any attempt to achieve a desired level of quality of service must be uniformly applied to the traffic inside a communication and computer network and should not be intended for any specific service or user. That is another challenge that needs to be carefully addressed and its solutions achieved [13].

Maintaining security and integrity of information is another continuing challenge. The threat of sensitive information passively or actively falling into unauthorized hands is very real. In addition, proactive and unauthorized attempts to gain access to secure databases are also very real. These issues need to be resolved to gain the confidence of consumers so that they may use the innovations in communications and computer networking technologies to their fullest [13].

## 17.6 Summary and Conclusions

This chapter discussed the fundamentals of communications and computer networks and the latest developments related to these fields. Communications and computer networks have witnessed tremendous growth and sophisticated improvements over the last several decades.

Computer networks are essentially resource sharing systems in which users share the transmission media and the switching nodes. These are used for exchanging information among users that are not necessarily connected directly. There has been a manifold increase in transmission rates of transmission media and the processing power of the switching nodes (which are essentially computers) has also been multiplied. The emerging computer networks are supporting communication of different types of services in an integrated fashion. All types of information, irrespective of type and source, are being transported in the form of packets (e.g., ATM cells). Resources are being allocated to users on a dynamic basis for better utilization. Wireless communication networks are emerging to provide worldwide connectivity and exchange of information at any time.

These developments have also posed some challenges. Effective traffic management techniques, meeting QoS requirements, and information security are the major challenges that need to be surmounted in order to win the confidence of users.

## References


1. Bae, J., and Suda, T., "Survey of traffic control schemes and protocols in ATM networks," *Proceedings of the IEEE,* Vol. 79, No. 2, February 1991, pp. 170–189.
2. Beyda, W., "Data communications from basics to broadband," Third Edition, 2000.
3. Black, U., "ATM: foundation for broadband networks," Prentice-Hall, Englewood Cliffs, NJ, 1995.
4. Black, U., "Emerging communications technologies," Second Edition, Prentice-Hall, Englewood Cliffs, NJ, 1997.
5. Chou, C., "Computer networks in communication survey research," *IEEE Transactions on Professional Communication,* Vol. 40, No. 3, September 1997, pp. 197–208.
6. Comer, D., "Computer networks and internets," Prentice-Hall, Englewood Cliffs, NJ, 1999.
7. Goodman, D., "Wireless personal communication systems," Addison-Wesley, Reading, MA, 1999.
8. Goralski, W., "Introduction to ATM networking," McGraw-Hill, New York, 1995.
9. Freeman, R., "Fundamentals of telecommunications," John Wiley & Sons, New York, 1999.
10. Ilyas, M., and Mouftah, H.T., "Performance evaluation of computer communication networks," *IEEE Communications Magazine,* Vol. 23, No. 4, April 1985, pp. 18–29.
11. Kleinrock, L., "The latency/bandwidth tradeoff in gigabit networks," *IEEE Communications Magazine,* Vol. 30, No. 4, April 1992, pp. 36–40.
12. Kleinrock, L., "ISDN-The path to broadband networks," *Proceedings of the IEEE,* Vol. 79, No. 2, February 1991, pp. 112–117.
13. Leon-Garcia, A., and Widjaja, I., "Communication networks, fundamental concepts and key architectures," McGraw Hill, New York, 2000.
14. Mukherjee, B., "Optical communication networks," McGraw-Hill, New York, 1997.
15. Partridge, C., "Gigabit networking," Addison-Wesley, Reading, MA, 1994.
16. Rappaport, T., "Wireless communications," Prentice-Hall, Englewood Cliffs, NJ, 1996.
17. Schwartz, M., "Broadband integrated networks," Prentice-Hall, Englewood Cliffs, NJ, 1996.
18. Shay, W., "Understanding communications and networks," Second Edition, PWS, 1999.
19. Stallings, W., "Local and metropolitan area networks," Sixth Edition, Prentice-Hall, Englewood Cliffs, NJ, 2000.
20. Stallings, W., "ISDN and broadband ISDN with frame relay and ATM," Fourth Edition, Prentice-Hall, Englewood Cliffs, NJ, 1999.
21. Stallings, W., "High-speed networks, TCP/IP and ATM design principles," Prentice-Hall, Englewood Cliffs, NJ, 1998.
22. Yuan, X., "A study of ATM multiplexing and threshold-based connection admission control in connection-oriented packet networks," Doctoral Dissertation, Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, Florida 33431, August 2000.

# 18

# Control with Embedded Computers and Programmable Logic Controllers

Hugh Jack
*Grand Valley State University*

Andrew Sterian
*Grand Valley State University*



## 18.1 Introduction

Modern control systems include some form of computer, most often an embedded computer or programmable logic controller (PLC). An embedded computer is a microprocessor- or microcontroller-based system used for a specific task rather than general-purpose computing. It is normally hidden from the user, except for a control interface. A PLC is a form of embedded controller that has been designed for the control of industrial machinery. (See Figure 18.1.)

A block diagram of a typical control system is shown in Figure 18.2. The controller monitors a process with sensors and affects it with actuators. A user interface allows a user or operator to direct and monitor the control system. Interfaces to other computers are used for purposes such as programming, remote monitoring, or coordination with another controller.

When a computer is applied to a control application, there are a few required specifications. The system must always remain responsive and in control of the process. This requires that the control software be real-time so that it will respond to events within a given period of time, or at regular intervals. The systems are also required to fail safely. This is done with thermal monitoring for overheating, power level detection for imminent power loss, or with watchdog timers for unresponsive programs.

## 18.2 Embedded Computers

An embedded computer is a microprocessor- or microcontroller-based system designed for dedicated functionality in a specialized (i.e., nongeneral-purpose) electronic device. Common examples of embedded computers can be found in cell phones, microwave ovens, handheld computing devices, automotive systems, answering machines, and many other systems.

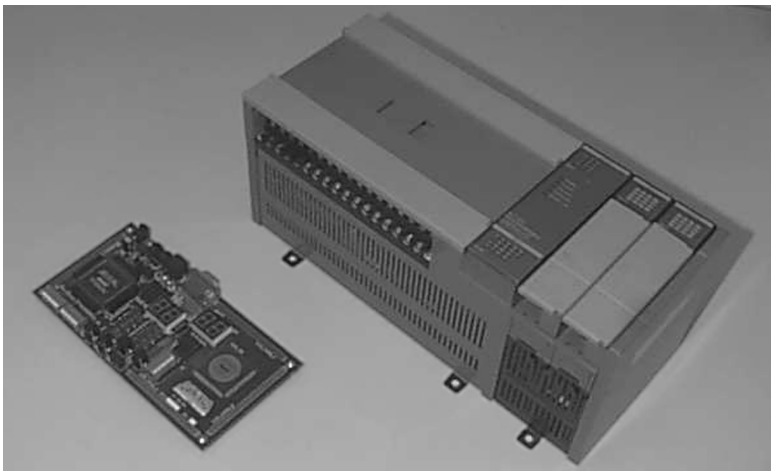


**FIGURE 18.1** An embedded computer with an Altera FPGA (front-left) and an Allen Bradley SLC500 programmable logic controller (top-right).

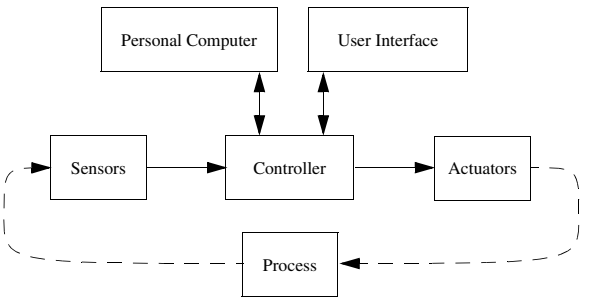



**FIGURE 18.2** An example block diagram of a computer controlled application.

The design constraints and parameters for an embedded computer are usually different from those of a general-purpose computer. Although the latter is designed for maximum computing power and support for the latest interconnection and peripheral standards, an embedded computer is designed to be just powerful enough and to support only the interfaces and protocols that are specifically required. The constraints of an embedded computer design often include size, power consumption and heat dissipation, and cost.

## Hardware Platforms

### Microcontroller-Based Systems

Microcontrollers are closely related to the microprocessors that power today's general-purpose computers. They differ from microprocessors, in general, by being highly integrated, with built-in peripherals that minimize total system part count, having low power consumption, providing a small amount of on-chip RAM and ROM, and having several general-purpose input/output (I/O) lines available for instrument sensors and control. For this reason, a microcontroller-based embedded system may be designed with very few external components. In contrast, a microprocessor-based system requires external RAM, external peripherals, and I/O interfaces, and often dissipates so much heat that active cooling is required for proper operation.

The peripherals built into many microcontrollers include serial-line interfaces (such as RS232), timers, pulse generators, event counters, etc. These peripherals support many sensor and actuator control functions.

For example, pulse generators and timers can be used to construct stepper motor drive sequences. Microcontrollers are becoming increasingly specialized with respect to the data communication interfaces they support. While many support the ubiquitous RS232, SPI, and $I^2C$ protocols, recent microcontrollers have built-in support for interfaces such as USB.

In order to minimize power consumption, most microcontrollers have a special sleep or standby mode in which no instructions are executed and very little power is consumed. Microcontrollers can be programmed to awaken in response to an external event so that the program code is executed, and power consumed, only when necessary.

Microcontrollers are a very large semiconductor market due to the wide range and high volume of devices that use them. There are many manufacturers and models of microcontrollers, ranging from tiny 8-pin devices with minimal functionality and costing mere pennies, to large devices with hundreds of pins, many features, and much higher cost. This broad spectrum reflects the highly specific nature of an embedded computer and its design.

### FPLD-Based Systems

Field-programmable logic devices (FPLDs) such as CPLDs (complex programmable logic devices) and FPGAs (field-programmable gate arrays) are a more recent alternative to microcontrollers for embedded computer design. An FPLD represents a programmable hardware device; the actual hardware functionality of the device is what is being designed. A microcontroller, in contrast, has fixed hardware functionality and is programmed with software. It is possible, however, to design an FPLD that behaves as a microcontroller, and is further programmed in software. The programmable hardware functionality, however, affords the designer a much greater degree of flexibility over a fixed hardware solution. The price for this flexibility, however, is complexity.

FPLDs may be designed from the ground up or may be composed of one or more predesigned core and peripheral blocks. It is possible, for example, to purchase microcontroller core functions, peripheral functions, etc. and assemble them to form a customized microcontroller on an FPLD with non-recurring engineering (NRE) costs that are much lower than a full custom chip design.

FPLDs can often be programmed "on-the-fly," allowing for reconfigurable computing. This is a computing paradigm that reprograms a system at the hardware level while it is in operation, according to system demands. This means, for example, that the same hardware device can implement multiple bus protocols, interfaces, or algorithms as needed, rather than requiring a larger and more expensive device that supports all of the necessary functions but only uses one at a time.

### Digital Signal Processing Systems

Digital signal processing (DSP) devices are in many ways similar to microcontrollers with respect to peripheral integration, power consumption, etc. but also have specialized hardware support for common DSP operations, such as filtering. DSP devices are ideal for use in systems that process speech and music, or for robust control and communications applications. The specialized hardware support of these devices means that they are capable of sustaining much higher effective computation rates (on signal processing tasks), but at the same clock speed and power dissipation as the more general-purpose microcontrollers.

### Real-Time Systems

Most embedded systems must operate in *real time*, that is, they must respond in a timely fashion to external events such as user commands and sensor readings. When an absolute upper limit on response time is required (and guaranteed), the system is a *hard real-time system*; otherwise, it is a soft real-time system. Systems that have safety constraints, such as automotive and industrial control systems, are often hard real-time systems so that absolute maximum time delays can be computed and verified for safety-critical events.

Real-time computation is effected using *interrupts*. These are mechanisms supported by all common microcontrollers that cause a change in the flow of execution of the program when the interrupt occurs. The program that is executed in response to the interrupt is expected to respond in some way to the interrupt.

For example, an interrupt may occur when a digital logic level changes at a device pin, indicating a sensor condition, or it may occur when the user presses a button on a keypad, indicating that an action is desired and is to be performed immediately.

The implementation of a real-time software system may either be custom designed or may make use of a commercial real-time operating system (RTOS). Since the design of an interrupt-driven real-time system has many potential pitfalls, the usage of a mature RTOS can greatly speed development time.

### Embedded Modules

The functionality of commercially available embedded computing modules has been steadily increasing. It is common to find powerful microcontrollers, an Ethernet interface, and basic Internet protocol support all combined in a very small form factor for less than $50 in single quantities. This level of integration can greatly speed development time for network-enabled control or remote sensing applications.

## Hardware Interfacing

### Mechanical Switches

Switches are easily interfaced to digital logic with a resistor as shown in Figure 18.3. The mechanical nature of the switch may lead to *bounce* or oscillation of the digital signal for a brief period during the switch opening/closing action. This bounce may be eliminated in the software or with a small amount of additional hardware.

### Analog Inputs

Analog inputs that indicate one of the two conditions can be interfaced to a digital logic input with a simple comparator (Figure 18.4). A threshold voltage is set with a resistor divider. The comparator generates a digital signal, which indicates whether the analog input voltage is above or below the threshold voltage. This approach can be used for sensors such as optical interrupters (for part counting, motor movement detection, etc.), temperature limit sensors, and many others.

When the analog voltage itself is of interest (as in, for example, temperature measurements), an analog-to-digital converter (ADC) can be used to provide either a serial or a parallel representation of the voltage with a precision ranging anywhere from 8 bits to 16 bits and above. A serial ADC may require as few as two digital I/O pins on a microcontroller for transferring data, while a parallel ADC requires at least as

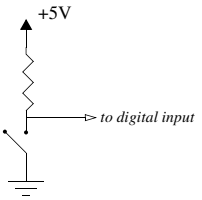



**FIGURE 18.3** A mechanical switch is easily interfaced to a digital input on a microcontroller using a single resistor.

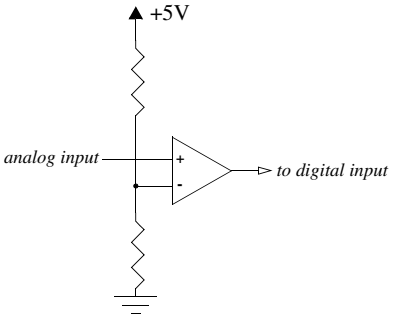



**FIGURE 18.4** A digital input driven by a comparator detects whether an analog voltage signal is above or below a threshold voltage (set with a resistor divider network).

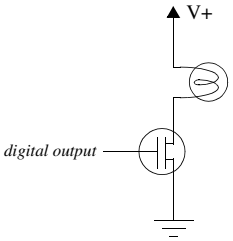



**FIGURE 18.5** A digital output can control a high- current device (such as a lamp pictured to the right) using a transistor as a switch.

many pins as there are bits of resolution, but can transfer an entire analog reading in one transaction for faster throughput.

Some microcontrollers have built-in ADC peripherals with multiple input channels enabling highly integrated low-cost analog sensor systems.

Analog voltages that are very small may be amplified with an instrumentation amplifier prior to analog-to-digital conversion. Instrumentation amplifiers have very high gain and high impedance, and hence, are suitable for sensors with very weak driving voltages and currents.

### Simple Actuators

Embedded computers are not usually capable of driving most practical actuators directly, since the latter often require voltages and currents not compatible with digital circuitry. As with sensors, however, some simple interface circuitry is all that is required. Simple on/off actuators such as lamps, LEDs, relay coils, etc. can be driven from a digital output, using a transistor as a switch, as shown in Figure 18.5. The digital output controls the on/off state of the transistor, which, in turn, either allows or does not allow current to flow through the actuator.

Motors can also be controlled using transistors as interfaces between digital outputs and the high-current motor coils. The lamp in Figure 18.5 can be replaced with a DC motor to allow simple on/off control of the motor. A set of four transistors arranged in an H-bridge configuration allows such a motor to rotate in either direction. Two H-bridge configurations can be used to control a stepper motor. In all cases, the speed and direction of rotation are under direct control of the embedded computer through its digital outputs.

### Analog Outputs

For actuators that require a variable analog voltage or current, a digital-to-analog converter (DAC) can be used as an interface between the embedded computer and the actuator. As with ADCs, DACs are available in a variety of bit widths, conversion speeds, number of channels, etc. Often, the current driving capacity of these devices is not sufficient and an additional buffer amplifier is required to meet the current demands of the actuator.

## Programming Languages

Embedded computers are most commonly programmed in low-level languages for maximum control over the hardware resources. The most time-critical sections of the code are generally programmed in assembly language, which is the lowest-level language understood by a microcontroller. The C language is generally used for higher-level structured programming. Even higher-level languages, such as C++ or Java, are not well suited for embedded programming as they require larger amounts of memory and are not designed for low-level access to hardware resources.

Figure 18.6 shows a fragment of an assembly language program written for the Microchip PIC 16F84A microcontroller. It enables power to a DC motor (through an external interface circuit) when two digital inputs are both at a logic 1 level. The code runs continuously, always checking for the status of the two digital inputs (which may be manual switches, current sensors, etc.).

The same code fragment written in C is shown in Figure 18.7. The code is more compact and easier to read since C is a higher-level language than assembly.

```
loop:
        btfss   PORTA,0         Check digital input bit 0 of Port A
        goto    turnoff             and disable motor if not 1
        btfss   PORTA,1         Check digital input bit 1 of Port A
        goto    turnoff             and disable motor if not 1

        bsf     PORTB,5         Enable motor by setting bit 5 of Port B
        goto    loop                and check inputs again

turnoff:
        bcf     PORTB,5         Disable motor by clearing bit 5 of Port B
        goto    loop                and check inputs again
```

**FIGURE 18.6** Fragment of assembly code for Microchip PIC 16F84A microcontroller. This code fragment examines two digital inputs (bits 0 and 1 of input Port A) and sets bit 5 of output Port B if both inputs are at a logic 1 level. The output can be used to enable or disable a DC motor with appropriate interface circuitry.

```
while (1) {
        if (PA0 && PA1) {// Check status of bits 0 and 1 in Port A
                PB5 = 1;      // Set bit 5 of Port B
        } else {
                PB5 = 0;      // Clear bit 5 of Port B
        }
}
```

**FIGURE 18.7** Fragment of C code to effect the same functionalilty as the code in Figure 18.6.

## 18.3 Programmable Logic Controllers

The modern programmable logic controller (PLC) is the successor of relay-based controls. The technological shift began in the 1960s, when the limitations of electromechanical relay-based controllers drove General Motors to search for electronic alternatives. The answer was provided in 1970 by Modicon, who provided a microprocessor-based control system. The programming language was modeled after relay ladder logic diagrams to ease the transition of designers, builders, and maintainers to these new controllers. Throughout the 1970s the technology was refined and proven, and since the early 1980s they have become ubiquitous on the factory floor.

Most PLC components are in card form that can be interchanged quickly in the event of a failure. A typical PLC application has about one hundred inputs and outputs, but the scale of the applications varies widely. A small PLC costing $200 might have six inputs and four outputs. A large application might involve multiple PLCs working together over an entire plant and collectively have tens of thousands of inputs and outputs. In general, the aggregated cost of PLC hardware per input and output is approximately $10–$50. This does not include the cost of sensors (typically $50–$100), actuators (typically $50–$200), installation (typically $10–$100), design, or programming.

Manufacturing control systems always require logical control and sometimes continuous control. Logical control involves the examination of binary inputs (on or off) from sensors and setting binary outputs to drive actuators. A simple example is a photosensor that detects a box on a conveyor and actuates an air cylinder to divert the box. Continuous control systems are used less frequently because of their higher costs and increased complexity. A typical continuous controller might use an analog output card ($1000) to output a voltage to a variable frequency motor driver ($1000) to control the velocity of a conveyor.

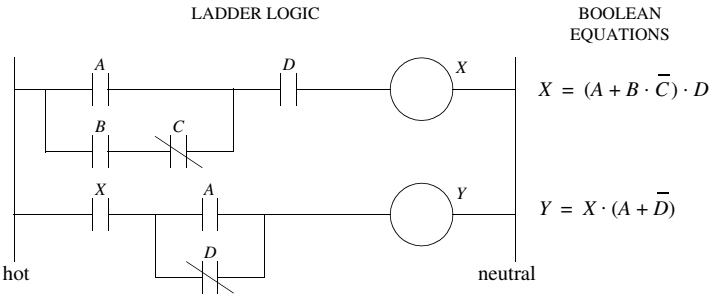



**FIGURE 18.8** A simple ladder logic program with equivalent Boolean equations.

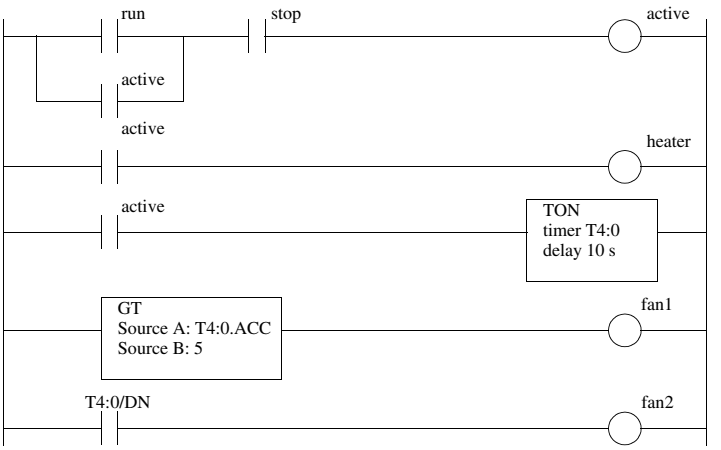



**FIGURE 18.9** A complex ladder logic example.

## Programming Languages

Every PLC can be programmed with ladder logic. Ladder logic uses input contacts (shown with two vertical lines) and output coils (shown with a circle). A contact with a slash through it represents a normally closed contact. In ladder logic, the left-hand rail is energized. When the contacts are closed in the right combinations, power can flow through the coil to the right-hand neutral rail.

Consider the ladder logic example in Figure 18.8. It is assumed that the *hot* rail at the left side has power, and the right side rail is *neutral*. When the contacts are opened and closed in the right combinations they allow power to flow through the output coils, thus actuating them. The program logic is interpreted by working from the left side of the ladder. In the first rung if $A$ and $D$ are on, the output $X$ will be turned on. This can also be accomplished by turning $B$ on, turning $C$ off, and turning $D$ on. In the second, the output $Y$ will be on if $X$ is on and $A$ is on, or $D$ is off. Notice that the branches behave as OR functions and the contacts in line act as an AND function. It is possible to write ladder logic rungs as Boolean equations, as shown on the right-hand side of the figure.

The example in Figure 18.8 contains only conditional logic, but Figure 18.9 shows a more complex example of a ladder logic program that uses timers and memory values. When the *run* input is active, output *heater* will turn on, 5 s later *fan1* will turn on, followed by *fan2* at 10 s. The first rung of the program will allow the system to be started with a normally open *run* push button input, or stopped with a normally closed push button *stop*. All stop inputs are normally closed switches, so the contact in this rung needs to be normally open to reverse the logic. The output *active* is also used to branch around the *run* to *seal-in* the run state. The next line of ladder logic turns on an output *heater* when the system is active. The

third line will run a timer when *active* is on. When the input to the *TON* timer goes on, the timer *T4:0* will begin counting, and the timer element *T4:0.ACC* will begin to increment until the delay value of 10 s is reached, at this point the timer done bit *T4:0*/*DN* bit will turn on and stay on until the input to the timer is turned off. The fourth rung will compare the accumulated time of the timer and if it is greater than 5 the output *fan1* will be turned on. The final rung of the program will turn on *fan2* after the timer has delayed 10 s.

A PLC scans (executes) a ladder logic program many times per second. Typical execution times range from 5 to 100 ms. Faster execution times are required for processes operating at a higher speed.

The notations and function formats used in Figure 18.9 are based on those developed by a PLC manufacturer. In actuality, every vendor has developed a different version of ladder logic.

### IEC 61131-3 Programming Languages

The IEC 61131 standards (formerly IEC 1131) have been created to unify PLCs [3,5]. The major portions of the standard are listed below.

IEC 61131-1 Overview
IEC 61131-2 Requirements and Test Procedures
IEC 61131-3 Data Types and Programming
IEC 61131-4 User Guidelines
IEC 61131-5 Communications
IEC 61131-7 Fuzzy Control

The most popular part of the standard is the programming specification, IEC 61131-3. It describes five basic programming models including ladder diagrams (LD), instruction list (IL), structured text (ST), sequential function charts (SFC), and function block diagrams (FBD). These languages have been designed to work together. It is possible to implement a system using a combination of the languages, or to implement the same function in different languages. A discussion of ST, SFC, and FDB programs follows.

#### *Structured Text*

A structured text program is shown in Figure 18.10. This program has the same function as the previous ladder logic example. The first line defines the program name. This is followed by variable definitions. The variables *run* and *stop* are inputs to the controller from sensors and switches. The variables *heater*,

```
PROGRAM example
VAR_INPUT
      run : BOOL ;
      stop : BOOL ;
END_VAR
VAR_OUTPUT
      heater : BOOL ;
      fan1 : BOOL ;
      fan2 : BOOL ;
END_VAR
VAR
      active : BOOL ;
      delay : TON ;
END_VAR
      active := (run OR active) & stop ;
      heater := active ;
      delay(EN := active, PRE := 10) ;
      IF ( delay.ACC > 5 ) THEN
                              fan1 := 1 ;
      ELSE
                              fan1 := 0 ;
      END_IF ;
      fan2 := delay.DN ;
END_PROGRAM
```

**FIGURE 18.10** A structured text program equivalent to Figure 18.9.

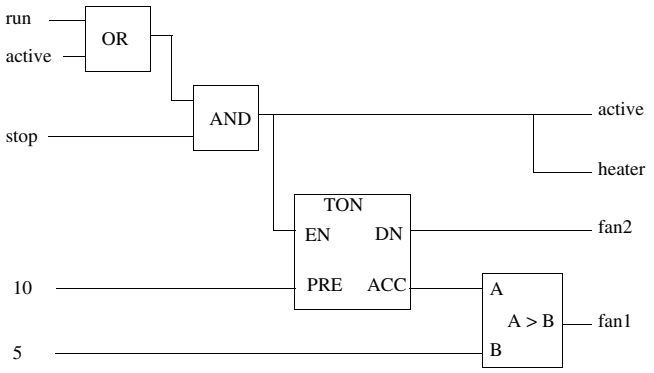



**FIGURE 18.11** A FBD program equivalent to Figure 18.9.

*fan1*, and *fan2* are outputs to the actuators in the system. The variables *active* and *delay* are internal to the program only.

The program section immediately follows the variable declarations. In the program the first two lines set the values of *active* and *heater*. The instruction *delay*(….) calls the instantiated timer. The argument *EN* := *active* sets the timer to run, and *PRE* := 10 sets the timer delay to 10 s. The following lines use an "if" statement to set the value of *fan1*, using the accumulated timer value *delay.ACC*. The value of *fan2* is then set when the timer accumulator has reached the delay time and set the done bit *delay.DN*.

Structured text is popular and shows potential for eventually replacing ladder logic as the most popular programming language.

***Function Block Diagrams***

A data flow model is the basis of function block diagrams. In these programs, the data flows from the inputs on the left to the outputs on the right. The example in Figure 18.11 is equivalent to the previous ladder logic example. The *OR* and *AND* functions are used to set the values of *active* and *heater*. The *TON* timer uses the enable *EN* and delay *PRE* inputs to drive the accumulator *ACC* and *DN* outputs. The *DN* output drives *fan2* while the *ACC* value is compared to the value of 5 to set the output *fan1*.

Data flow diagrams can be very useful for doing a high-level design of a control system.

***Sequential Function Charts***

An SFC is used to describe a system in terms of *steps* and *transitions*. A step describes a mode of operation or state in which some *action* is performed, normally setting outputs. Transitions determine the change of states, normally by examining inputs. (Note: Some readers may notice that SFCs are based on Petri nets.)

Figure 18.12 shows an example of an SFC to control storage tanks. When the controller is started and the *power* input goes true, it will empty the tanks. After that the *run* input will start cycles where both the tanks are filled and then emptied repeatedly.

In this example, the flow of control begins at the initial step *start*, and then moves to step *S1*. The action associated with the step is *R*, which will reset, or turn off the outputs *in_valve1* and *in_valve2*. The system will remain in step *S1* until the transition is fired by input *power*. After this there are two possible paths. If *empty1* and *empty2* are both true, the left-hand branch will be followed, otherwise the right-hand transition will fire and that branch will be followed. The left-hand branch sets the *run_light* on (with *S*), and turns off the *outlet_valve*. The right-hand branch will turn on *outlet_valves* until the inputs *empty1* and *empty2* are both on. At that point *run_light* will be turned on, and the *out_valves* turned off. Regardless of which branch was followed, the flow of execution will pause at the following transition until the input *run* becomes true.

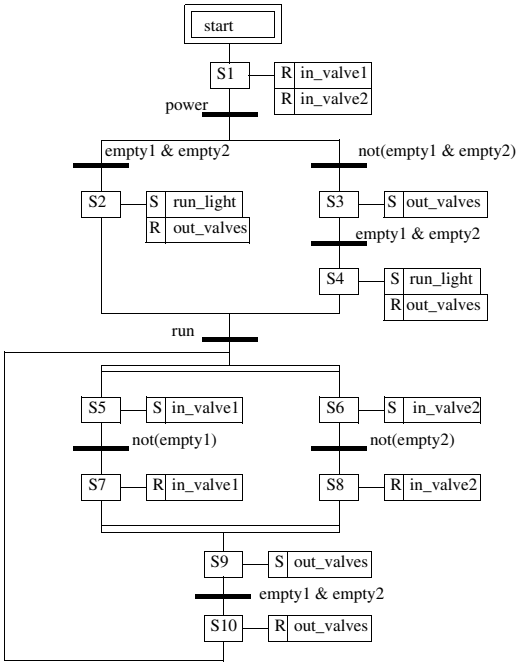



**FIGURE 18.12** An SFC program for tank level control.

After the *run* transition is fired the flow of execution splits into both the left and right branches, as indicated by the two horizontal lines. The left branch fills one tank, while the right branch fills the other tank independently. When both branches are complete the flow of execution rejoins at the second set of horizontal lines, and then activates step *S9*. After step *S10* the flow of execution returns to the point after the *run* transition.

The SFC programming method differs from other programming methods in that the program is not expected to run completely in a single scan, while all others must run completely in each scan.

## Interfacing

The installation and interfacing requirements for PLCs are driven by the need to protect people and equipment by failing safely. A typical wiring diagram for a PLC application is shown in Figure 18.13. At the top of the diagram a transformer is used to step down a higher supply voltage. This is immediately followed by a power disconnect and fuses. The power is then split into left and right rails, much like the ladder diagrams discussed earlier. Line 10 shows a master power control for the system. This includes a normally open start button and a normally closed stop button. These switches control a master control relay (MCR) *C1*. Notice that if power is supplied to the coil *C1*, it will close the contacts *C1* on the same run and hold *C1* on until the stop button is pushed. Another set of contacts is used on the left rail to disconnect power from the inputs to the PLC and the DC power supply. This control circuitry external to the PLC is required so that the stop buttons of a control system are able to directly disconnect the power. This is often required by law.

In this example the PLC is powered with 120 V AC, connected between the power rails. There are two 120 V AC inputs from normally open push buttons. The 24 V DC power supply is input to the *V+* on the output terminals of the PLC, which will then switch output power to solenoid *S1* and indicator light *L1*.

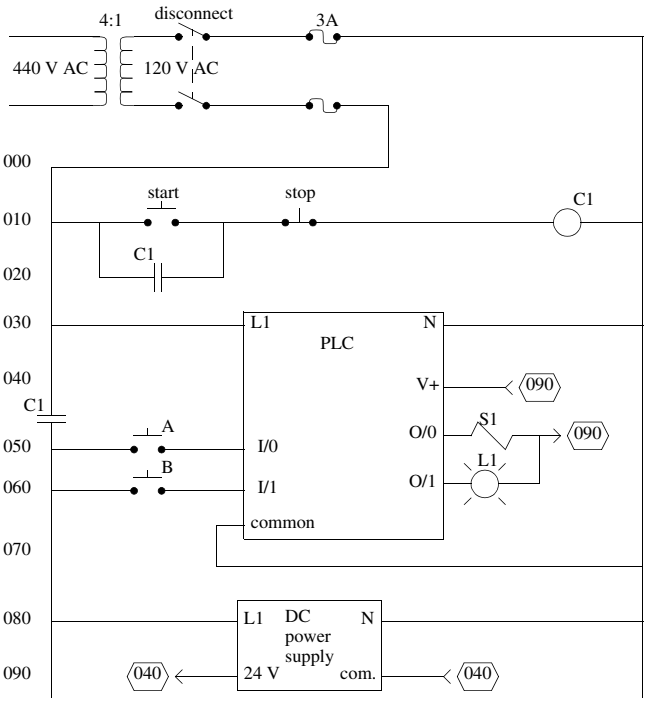



**FIGURE 18.13** A PLC wiring example.

## Advanced Capabilities

PLCs are often used in applications that go beyond basic logic solving. Some advanced programming and input/output (I/O) functions are listed below.

Calculations—The ability to do basic scientific calculations. Lower end PLCs only use integer math, while higher end PLCs also provide floating point math.
Analog I/O—Continuous voltage and current values can be input and output.
Feedback control—Proportional integral derivative (PID) controller calculations are provided as function blocks and can be used with analog I/O.
Communications—The ability to transmit data as strings over serial ports or to transfer parts of the PLC memory using proprietary protocols.
ASCII strings—Functions to manipulate ASCII strings.
System—Fault detection, status monitoring, interrupt routines, etc.
Fuzzy logic—Some PLCs include fuzzy set functions for nonlinear control problems.

At a minimum PLCs use communications for programming. But in many applications PLCs are used to communicate with other devices. In the past, most communications were based on proprietary, or closed, standards. More recently a few open communication standards have been developed and are supported by many vendors; these include Profibus, DeviceNet, CanBus, and ModBus. There has also been a trend to use more universal communication standards such as RS-232, RS-422, RS-485, and Ethernet. An example of an automation system is shown in Figure 18.14. An RS-232 connection is used between a laptop computer (e.g., COM1) and PLC1 for programming. DH+ is used to connect PLC1, PLC2, and the HMI; it is a proprietary communication standard developed by Allen-Bradley. An operator can use the Human Machine Interface (HMI) to display data and accept operator input and communicate these values directly to both PLCs. Devicenet, an open automation standard, is used to connect PLC2 to a welding controller.

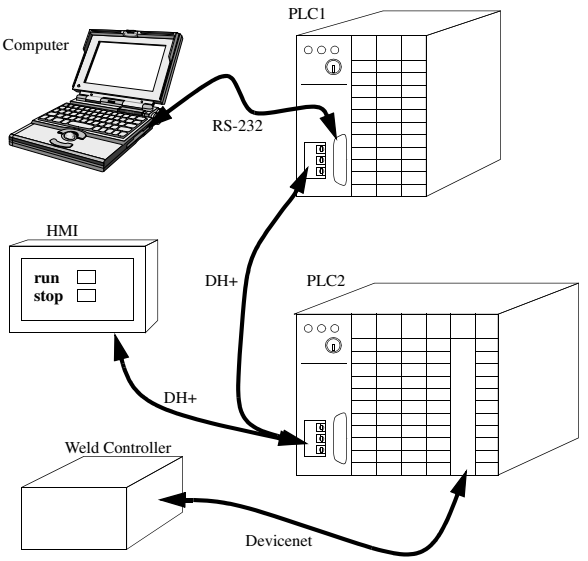



**FIGURE 18.14** PLC communication example.

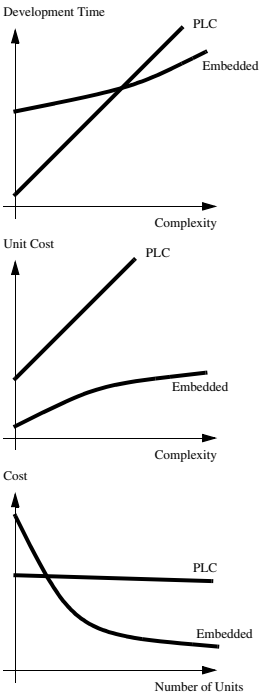



**FIGURE 18.15** Relative trade-offs between control solutions.

## 18.4 Conclusion

PLCs and embedded controllers are complementary technologies and, when applied strategically, they will both provide low cost and reliable solutions to control problems. Figure 18.15 shows the relative trade-offs between the controllers. In general, an embedded controller requires more initial development time than a PLC for a simple system. As the system grows more complex, the embedded controller benefits

from the existence of software libraries and design tools. When using a PLC the cost of the purchased hardware will always be higher per unit. The development costs for an embedded computer will usually be higher, but these become minimal when amortized over a large number of units. As a result, embedded controllers are typically selected for applications that will be mass-produced and allow a greater development time, such as a toy robot. PLCs are often selected for applications that only require a few controllers and are to be completed in a relatively short time, such as the production machines to make a toy.

## References

1. Bryan, L.A., Bryan, E.A., *Programmable Controllers,* Industrial Text and VideoCompany, 1997.
2. Filer, R., Leinonen, G., *Programmable Controllers and Designing Sequential Logic,* Saunders College Publishing, 1992.
3. Lewis, R.W., *Programming Industrial Control Systems using IES1131-3,* The Institution of Electrical Engineers, 1998.
4. Petruzella, F., *Programmable Logic Controllers,* Second Edition, McGraw-Hill, 1998.
5. *Programmable Controllers—Part 3: Programming Languages,* IEC 61131-3 Ed. 1.0, 1993.
6. Stenerson, J., *Fundamentals of Programmable Logic Controllers, Sensors and Communications,* Prentice-Hall, 1998.
7. Webb, J.W., Reis, R.A., *Programmable Logic Controllers, Principles and Applications,* Prentice-Hall, 1995.

# 19

# Introduction to Data Acquisition

Jace Curtis
*National Instruments, Inc.*

The purpose of a data acquisition system is to capture and analyze some sort of physical phenomenon from the real world. Light, temperature, pressure, and torque are a few of the many different types of signals that can interface to a data acquisition system. A data acquisition system may also produce electrical signals simultaneously. These signals can either intelligently control mechanical systems or provide a stimulus so that the data acquisition system can measure the response. A data acquisition system provides a way to empirically test designs, theories, and real world systems for validation or research. Figure 19.1 illustrates a typical computer-based data acquisition module.

The design and the production of a modern car, for instance, relies heavily on data acquisition. Engineers will first use data acquisition to test the design of the car's components. The frame can be monitored for mechanical stress, wind noise, and durability. The vibration and temperature of the engine can be acquired to evaluate the design quality. The researchers and engineers can then use this data to optimize the design of the first prototype of the car. The prototype can then be monitored under many different conditions on a test track while information is collected through data acquisition. After a few iterations of design changes and data acquisition, the car is ready for production. Data acquisition devices can monitor the machines that assemble the car, and they can test that the assembled car is within specifications.

At first, data acquisition devices stood alone and were manually controlled by an operator. When the PC emerged, data acquisition devices and instruments could be connected to the computer through a serial port, parallel port, or some custom interface. A computer program could control the device automatically and retrieve data from the device for storage, analysis, or presentation. Now, instruments and data acquisition devices can be integrated into a computer through high-speed communication links, for tighter integration between the power and flexibility of the computer and the instrument or device.

Since data acquisition devices acquire an electric signal, a transducer or a sensor must convert some physical phenomenon into an electrical signal. A common example of a transducer is a thermocouple. A thermocouple uses the material properties of dissimilar metals to convert a temperature into a voltage. As the temperature increases, the voltage produced by the thermocouple increases. A software program can then convert the voltage reading back into a temperature for analysis, presentation, and data logging. Many sensors produce currents instead of voltages. A current is often advantageous because the signal will not be corrupted by small amounts of resistance in the wires connecting the transducer to the data acquisition device. A disadvantage of current-producing transducers, though, is that most data acquisition devices measure voltage, not current. Generally, the data acquisition devices that can measure current use a very small resistance of a known value to convert the known current into a readable voltage. Ultimately, the device is then still acquiring a voltage.

**FIGURE. 19.1**

Analog signals for data acquisition can be grouped into two basic classes: random and deterministic. Data acquisition devices can both acquire and generate these types of signals. Random signals never repeat and have a flat frequency spectrum. Microphone static is an example of a random signal. A deterministic signal, unlike random signals, can be represented by a sum of sinusoids. Deterministic signals can be subdivided into periodic and transient signals. Periodic signals constantly repeat the same shape at regular intervals over time, while transient signals start and end at a constant level and do not occur at regular intervals. Transient signals are nonperiodic events that represent a finite-length reaction to some stimulus.

Digital input and output are commonly incorporated into data acquisition hardware for sensing contacts, controlling relays and lights, and testing digital devices. The most commonly used digital levels are TTL and TTL-compatible CMOS. These are both very common 5-V standards for digital hardware. Digital transfer rates to and from the data acquisition hardware vary from unstrobed to high speed. Unstrobed digital input and output involves setting digital lines and monitoring states by software command. This form of digital input and output is also known as static or immediate digital I/O. The maximum speed of an unstrobed I/O is highly dependent on the computer hardware, the operating system, and the application program. Pattern digital I/O refers to inputs and outputs of digital patterns under the control of a clock signal. The speed at which the data can be sent or received depends on the amount of data, the characteristics of the data acquisition hardware, and the computer speed.

The final type of I/O on computer-based data acquisition hardware is counter/timer I/O. Counter/timers are capable of measuring or producing very time-critical digital pulses. These pulses, like the digital input and output, are generally TTL or TTL-compatible CMOS. These components are used for measuring or producing a number of time-critical signals including event counting, pulse train generation, frequency-shift keying, and monitoring quadrature encoders. The two main characteristics of a counter/timer are the counter size and maximum source frequency. The counter size is generally represented in bits and determines how high a counter can count. For instance, a 32-bit counter can count $2^{32} - 1 = 4{,}294{,}967{,}295$ events before it returns the count value back to zero. The maximum source frequency represents the speed of the fastest signal the counter can count. An 80-MHz counter can count events that are as fast as 12.5 ns apart. An "event" is actually the rising or falling edge of a digital signal.

No real situation will ever have perfect signals or be completely free of noise. Signal conditioning is a method to remove, as much as possible, unwanted components of a digital or analog signal. A real analog signal usually comprises both deterministic and random signals, and a digital signal is not going to be perfectly square. Measurement hardware, particularly for high-frequency analog signals, is usually equipped with an antialiasing filter. This is a low-pass filter that blocks frequencies above the desired frequency range and increases the accuracy of the measurements. Digital and counter/timer lines are also commonly fitted with filters that remove spikes from the signal that could otherwise be mistakenly

counted as a rising or falling edge. Isolation is another type of signal conditioning that separates the measurement hardware circuitry from the signal being measured. This is done to remove large differences in electric potential between the measurement hardware and the signal, and it protects the measurement hardware from damage, given a large surge in voltage or current.

The heart of a data acquisition device is a digital-to-analog converter (DAC), an analog-to-digital converter (ADC), or some combination of the two. An ADC has a finite list of values which represents voltages. The purpose of the ADC is to select a value from this list, which is closest to an actual voltage at a specified time. The value is then transferred in binary format to a computer. Alternatively, a DAC can produce an analog voltage from a list of binary values. The voltage generated by a basic DAC stays the same until it receives another value from the computer. In order to acquire and produce analog waveforms, the DAC and ADC must activate at precise intervals. Consequently, measurement hardware has timing circuitry to produce a pulse train of a constant frequency to control the ADC and DAC.

The data that is transferred from the ADC and to the DAC travels to the computer over a bus. A bus is a group of electrical conductors that transfer information inside a computer. Some common examples of a bus are PCI and USB. The bus can carry both control information and binary measurement data to and from measurement hardware. One of the most important considerations in selecting a bus is bus transfer rate, usually expressed in megabytes per second (Mbytes/s). A single analog value could require less than 1 byte or as much as 4 bytes, depending on the type of measurement hardware. The bus is shared among multiple devices, so data acquisition devices often have on-board memory to serve as a holding place for data when the bus is not available. In very fast data acquisition routines, the memory can hold all the data, and at the end of the acquisition, all the data can be transferred to the computer for processing.

When data is acquired at high speeds on multiple channels, it is often important to understand the phase relationship from one signal to the next. If the signals are generated or acquired on multiple data acquisition devices, there are a number of ways to synchronize the systems and preserve relative phase relationships. One way is to share the ADC and DAC clock between the data acquisition devices. The real-time system integration bus (RTSI) is a bus that can connect multiple devices together to share timing circuitry among multiple devices. Phase-lock looping (PLL) is a more sophisticated synchronization method. A reference signal is supplied to all the data acquisition devices, and the internal clocks stay in phase with the reference signal. Consequently, the phase relationship can be reserved even if different measurement hardware is using different sampling or update speeds. Figure 19.2 is a diagram showing the components of a typical data acquisition hardware.

The difference between an actual analog voltage and the closest voltage from the list of binary values is called the quantization error. In a perfect digitizing measurement system free of noise, the quantization error would solely explain any difference between the actual voltage and the measured voltage. No measurement hardware and no environment, however, are perfect. The accuracy of an instrument describes the amount of uncertainty when considering quantization error, unavoidable system noise, and hardware imperfections. Accuracy is sometimes confused with precision. Precision refers to the amount of deviation in multiple measurements connected to a constant and level signal source. Even if an instrument is

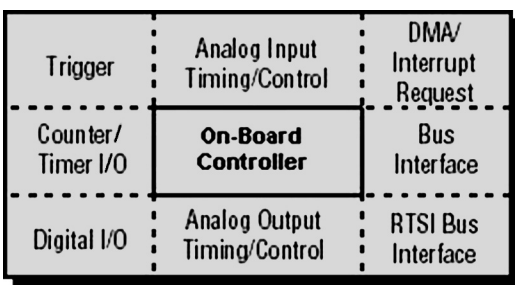



**FIGURE 19.2**

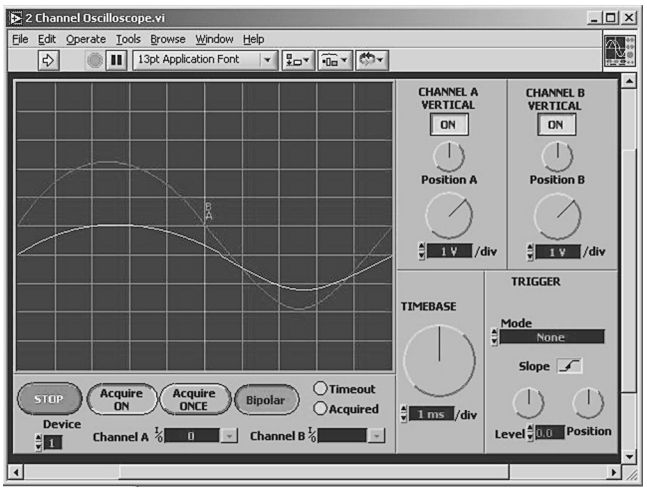



**FIGURE 19.3**

precise, it could still be inaccurate if the readings were consistent but significantly different than the actual value of the signal.

The accuracy of a data acquisition system can change with temperature, time, and usage. Data acquisition hardware can store on-board correction constants for offset and gain errors. An offset error is a constant difference between the measured and actual voltage, regardless of the voltage level. A gain error increases linearly as the measured voltage increases. Some data acquisition hardware also include an accurate voltage source on-board that can be periodically used as a reference to correct the gain and offset error parameters.

The final piece of a data acquisition system to understand is the software. The driver software is a set of commands that a programmer can incorporate into a program. The driver software is usually supplied by the manufacturer of the hardware and can be used in a variety of programming languages. A programmer can use a programming language to build an application from the driver software like the one in Figure 19.3. The application is then ready for an end user to easily control and acquire data from the hardware—a custom instrument built specifically for the user's needs.

# 20

# Computer-Based Instrumentation Systems

Kris Fuller
*National Instruments, Inc.*



Today's computer-based and networked measurement and automation systems contain powerful software that brings high-performance in a familiar environment. By using these systems, engineers lower their costs while increasing productivity and create more customized solutions that directly match their needs.

Electrical and electronics test instruments have always borrowed from contemporary technology that was widely used elsewhere. The jeweled movement of the nineteenth century used in clocks was first adapted to build analog meters. In the 1930s, when the variable capacitor, variable resistor, and vacuum tubes began to be widely accepted pieces of the radio, the first electronic instruments were introduced using the same components. As display technologies were improved for use on the first televisions, oscilloscopes and analyzers began using the same technology to display the user's measurements (see Figure 20.1). These first steps toward computer-based instrumentation met significant challenges. Computerized ins-trument systems of the 1960s required custom hardware interfaces and low-level assembly languages. The development of standards, such as the introduction in 1976 of the general-purpose interface bus for instrument-to-computer connections, provided the foundation for revolutionary improvements in the development and use of computer-based instruments.

Using the general-purpose interface bus, engineers began writing programs, first in BASIC, then C-based languages, and ultimately graphical development environments, that transformed their computers into efficient instrument controllers that also had the capability of electronically storing data. In the 1980s, digitizers and computer plug-in boards for data acquisition became widely accepted alternatives to expensive standalone instruments. With this combination of software and hardware, engineers began creating "virtual instruments."

Throughout the 1980s and 1990s, the idea of virtual instruments gained wider acceptance as the power of desktop computers increased exponentially. First consumer and then corporate demand for faster, more efficient CPUs, more capable and compact ASICs, faster and larger hard drives, and more capable interface buses played right into the hands of those designing computer-based instrumentation systems.

Today's instrumentation systems are being greatly influenced by the personal computer and Internet revolutions. Personal computers are now equipped with powerful computational engines that can be combined with software to create a sophisticated measurement instrument. The data that are acquired by the computer-based instrumentation system can then be easily transferred to anyone anywhere in the world who is connected to the instrumentation machine via the Internet.

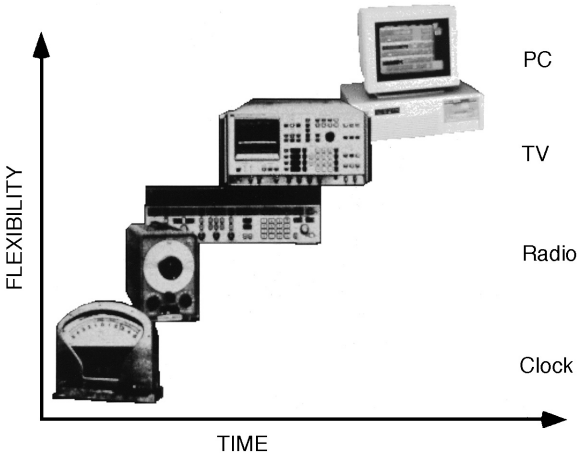



**FIGURE 20.1** History of instrumentation.

With advanced software and fast processing and data transfer speeds, computers could recreate the input/output, signal conditioning, storage, digitizing, and analysis capabilities of traditional instruments, using data acquisition (DAQ) and other measurement devices. Moreover, these computer-based devices could experience continuous performance improvements at a much lower cost than traditional standalone instruments as the overall technology of the computer industry continually advanced. Using computer-based tools, engineers could update their instrumentation systems by simply buying a new computer.

To understand the flexibility of computer-based instrumentation, it is instructive to examine more closely the individual components of these systems. First let us examine the software.

## 20.1 The Power of Software

Modern computer-based instrumentation involves a sophisticated mix of software, from instrument drivers to development software to test management software that integrate seamlessly for a complete enterprise solution. Software lies at the heart of computer-based instrumentation systems. They provide the flexibility and interoperability to provide custom solutions for each application. Each scientist or engineer can use the software to customize the particular test application to meet individual needs. Using traditional instrumentation, the vendor defined the functionality, interface, and parameters of each instrument. These limitations are no longer as prevalent. Scientists and engineers can now define and change the functionality of instrumentation by updating the software or by changing individual hardware components. The interface and parameters the operator interacts with can also be customized to meet individual needs by manipulating the user interface created in the software.

Software used in today's computer-based instrumentation systems reside in one of three levels. Instrument drivers lie at the primary level of computer-based software architecture. Instrument drivers interact with hardware, pulling measurement data into the computer. They also encapsulate the code that handles the command creation and communication functions required to communicate across GPIB with standalone instruments. Instrument drivers make test systems more maintainable because they can be easily upgraded or changed whenever hardware changes are made.

The second level of the software architecture is made up of development software, which can build the graphical user interface (GUI) for a particular instrument or measurement device. The application software layer is also where the specific personality of the computer-based instrument is defined. The customization available at this level depends on the software tool being used. Some software is written for specific hardware, and the functionality, look, and personality are primarily fixed. More open development environments exist and allow the user to greatly customize the instrumentation system. National Instruments LabVIEW™ and Measurement Studio™ and Microsoft Visual Basic and Visual C++ are

examples of such development environments. Using either graphical-based programming or the traditional C-based languages, engineers can customize their measurement interfaces so that they contain only the knobs, switches, and graphs they need for performing high-level analysis and data display.

Since many different options exist when considering application software, it is important to understand the primary uses for this software. Choose a development environment that can be used to increase productivity and fulfill all requirements of the measurement application. National Instruments LabVIEW, Microsoft Visual Basic, and Microsoft Visual C are the most used software packages for instrumentation applications. National Instruments LabVIEW is a graphical development environment that has specific functions for many common measurements and rapid user interface development. Microsoft Visual Basic and Visual C are text-based languages popular in all types of software applications. To fully take advantage of these text-based development environments, measurement focused add-ins are available from several companies.

The third level of the software architecture is typically made up of a test executive, which manages the sequence or order that tests execute. They support several popular test codes and can generate reports using many mainstream software programs such as Microsoft Excel. These off-the-shelf test packages can reduce the cost of testing products while freeing engineers to concentrate on the kinds of tests they are running. They no longer need to spend valuable time building proprietary software that carries out test management functions.

## 20.2 Digitizing the Analog World

Before software can even begin to analyze measurement data, it must be converted from the analog world into the computer's way of thinking. This involves numerous types of hardware that digitize analog signals. As explained previously, when acquiring these signals from standalone instrument hardware, the instrument driver software arranges the data so that development software can perform an additional level of data analysis and presentation. However, other hardware such as plug-in measurement devices, programmable logic controllers (PLCs), distributed I/O systems, and devices that support USB, IEEE-1394, and serial communication can also acquire signals using their own set of driver software (see Figure 20.2).

As noted previously, with today's technology, computer-based hardware can acquire signals at rates that often match or exceed that of standalone instruments, and computer-based measurement devices can digitize these signals at speeds and resolutions that rival traditional instruments. However, with computer-based technology, hardware integration with software produces some key differences. With this powerful combination, engineers can create instrumentation that specifically meets their needs. They define the features instead of an instrument vendor making those decisions for them. Once more, with computer-based devices, engineers can easily send data from their instruments across the Internet, using Ethernet and other standard computer-based technologies.

This flexibility, of course, brings choices. When choosing among the different types of measurement hardware, users must keep in mind their purpose or application. Acquiring high-speed signals, say up to 100 MHz, or rapidly changing signals, such as with sound or vibration analysis, can be done with digitizers and dynamic signal analyzers that plug-in to the computer via the PCI or PXI/CompactPCI bus. Making the right choice depends on the application.

Plug-in data acquisition boards have the advantage of being able to acquire data directly into the computer memory at very high speeds. One disadvantage to plug-in boards is that, by themselves, they do not provide an industrial interface for all transducers and signals. For example, thermocouples generate very small signals, and therefore their signals need to be linearized and require cold-junction compensation. Other applications may require particular signal conditioning features, like multiplexing to increase channel count, excitation for proper behavior, filtering to remove noise, isolation to protect the system from high voltage signals, or amplification for low-voltage signals. A specialized signal conditioning add-on is usually used in conjunction with a data acquisition board to fulfill the needs of connecting the computer to the external world.

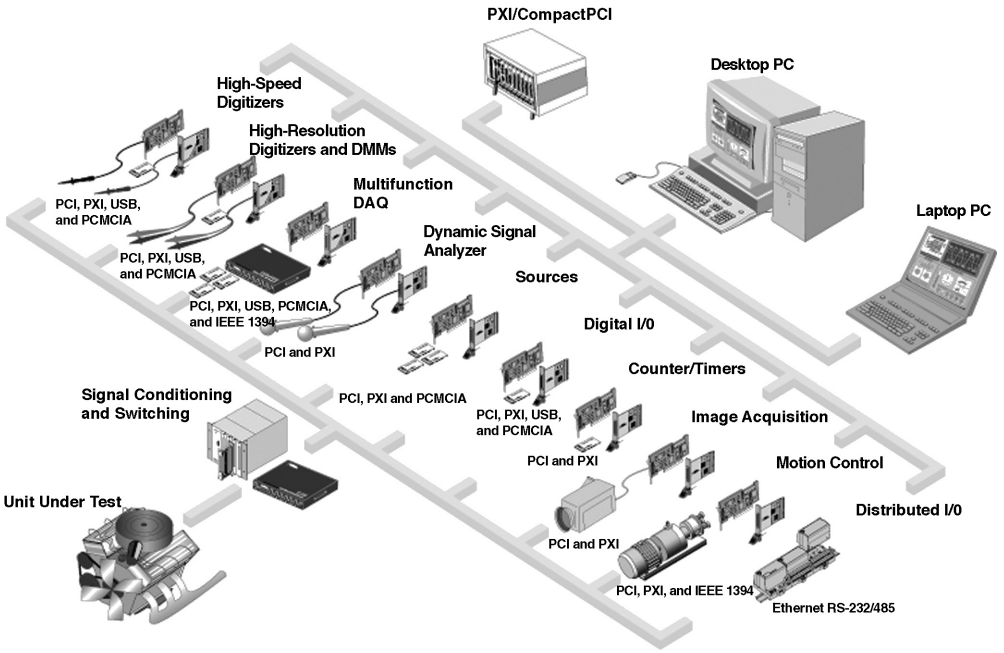



**FIGURE 20.2**

Some applications are in areas where having computers located adjacent to the measurements is not plausible. Distributed input–output devices are available for these types of applications. The digitizing and conditioning hardware is connected back to the computer via a serial, RF, or Ethernet connection.

Various types of measurements can be brought into the computer and sent from the computer in today's measurement and automation applications. Image acquisition devices and motion controllers can also be integrated into most measurement applications.

## 20.3 A Look Ahead

Today's software incorporates the power of the Internet and standard communication protocols, making the development of networked measurement applications simple, whether collecting data across the Web or publishing data via a Web browser.

This integration of the Web opens the door to new levels of software architecture that include entire enterprises. Engineers and scientists increasingly understand that acquiring and analyzing data is only a part of the equation. Data from these tests can be stored in central databases where it can be shared with peers. This sharing or management of data can lead to powerful efficiencies. Re-analysis of tests, comparisons of tests, and combinations of data can lead to new understandings and better decision-making.

Setting up these management systems requires a web of data collection, data repository, and reporting and analysis systems—all of which can be set up on computers.

Equally powerful changes are occurring with computer-based hardware. Embedded processors can now run real-time operating systems that perform deterministic tasks independently of the computer's CPU.

Developing real-time applications now can occur on a familiar desktop computer and Microsoft Windows-based environment. It opens a whole wealth of applications to engineers who are not steeped in the intricacies of traditional real-time programming but need the demanding performance that it delivers.

The instrumentation industry has always leveraged common available technologies to create power test equipment. Computer-based technologies and the Internet have enabled the measurement and automation community to create more customized solutions that are more connected to the everyday world than ever before.

# 21

# Software Design and Development*




Margaret H. Hamilton
*Hamilton Technologies, Inc.*


A software-based system can be neatly compared with a biological entity called a superorganism. Comprising software, hardware, peopleware and their interconnectivity (such as the Internet), and requiring all to survive, the silicon superorganism is itself a part of a larger superorganism—for example, a medical system including patients, drugs, drug companies, doctors, hospitals, and health care centers; a space mission including the spacecraft, the laws of the universe, mission control, and the astronauts; a system for researching genes including funding organizations, funds, researchers, research subjects, and genes; a financial system including investors, money, politics, financial institutions, stock markets, and the health of the world economy; or it could be just the business itself.

Whether that business be government, academic, or commercial, the software-based system, like its biological counterpart, must grow and adapt to meet rapidly changing requirements. And, like other organisms, the business has both physical infrastructure and operational policies, which guide and occasionally constrain its direction and the rate of evolution, which it can tolerate without becoming dysfunctional.

Compared to a biological superorganism, which may take many generations to effect even a minor hereditary modification, software can be modified immediately. This makes it far superior in this respect to the biological entity in terms of its evolutionary adaptability. Continuity of business rules and/or the physical infrastructure provides a natural tension between "how fast the software can change" and "how rapidly the overall system can accept change." Software, the brain of the silicon superorganism, controls the action of the entire entity. Keep in mind, however, it was a human being that created the software.

In this chapter we will discuss the tenets of software, what it is and how it is developed, as well as the precepts of software engineering, which are the methodologies by which ideas are turned into software.

* Parts of this chapter were taken from *Object Thinking: Development Before the Fact*, M. H. Hamilton and W. R. Hackler, in press.

## 21.1 The Notion of Software

Software is the embodiment of logical processes, whether in support of business functions or in control of physical devices. The nature of software as an instantiation of process can apply very broadly, when modeling complex organizations, or very narrowly as when implementing a discrete numerical algorithm. In the former case, there can be significant linkages between reengineering businesses to accelerate the rate of evolution—even to the point of building operational models, which then transition into application suites and thence into narrowly focused implementations of algorithms, as above. Software thus has a potentially wide range of application, and that well designed has a potentially long period of utilization.

While some would define software as solely the code that a programming language generates from the compilation process, a broader and more precise definition includes requirements, specifications, designs, program listings, documentation, procedures, rules, measurements, and data as well as the tools used to create, test, optimize, and implement the software.

That there is more than one definition of software is a direct result of the confusion about the very process of software development itself. A 1991 study by the Software Engineering Institute (SEI) [1] amplifies this rather startling problem. SEI developed a methodology for classifying an organization's "software process maturity" into one of five levels which range from Level 1, the initial level (where there is no formalization of the software process), to Level 5, the optimizing level where methods, procedures, and metrics are in place with a focus toward continuous improvement in software reliability. The result of this study showed that fully 86% of organizations surveyed in the United States were at Level 1 where the terms "ad-hoc," "dependent on heroes," and "chaotic" are commonly applied. And, given the complexity of today's Internet-based applications, it would not be surprising to see the percentage of Level 1 organizations increase.

Creating order from this chaos requires an insightful understanding into the component parts of software as well as the development process. Borrowing again from the world of natural science, an entelechy is something complex that emerges when a large number of simple objects are put together. For example, one molecule of water is rather boring in its utter lack of activity. But pour a bunch of these molecules into a glass and there is a ring of ripples on the water's surface. If many of these molecules are combined, the result is an ocean. So too software. By itself, a line of code is a rather simple item. But combine many lines and the result is a complex program. Add additional programs and the result could be a system that can put a person on the moon.

Although the whole is indeed bigger than the sum of its parts, one must still understand those parts if the whole is to work in an orderly and controlled fashion. Like a physical entity, software can "wear" as a result of maintenance, changes in the underlying system, and updates made to accommodate the requirements of the ongoing user community. Entropy is a significant phenomenon in software, especially for Level 1 organizations.

Software at the lowest programming level is termed source code. This differs from executable code (i.e., which can be executed by the hardware to perform one or more specified functions) in that software is written in one or more programming languages and cannot, by itself, be executed by the hardware. A programming language is a set of words, letters, numerals, and abbreviated mnemonics, regulated by a specific syntax, used to describe a program to a computer. There are a wide variety of programming languages, many of them tailored for a specific type of application. C, one of today's more popular programming languages, is used in engineering as well as business environments while object-oriented languages such as C++ [2] and Smalltalk have been gaining acceptance in both of these environments. More recently, Java [3] has been gaining acceptance. In fact, it has become a language of choice for Internet-based applications. In the recent past, engineering applications have often used programming languages such as FORTRAN, HAL (or HAL/s) for NASA space applications, and Ada for government applications while commercial business applications have favored COBOL (COmmon Business Oriented Language). For the most part one finds that in any given organization there are no prescribed rules, other than those related to what is most popular at the moment, which dictate which languages are to be used. And, as one might expect, a wide diversity of languages is being deployed.

The programming language, whether it be C++, Java, Visual BASIC, C, FORTRAN, HAL/s, COBOL, or something else, provides the capability to code such logical constructs as that having to do with:

- *User Interface*. Provides a mechanism whereby the ultimate end-user can input, view, manipulate, and query information contained in an organization's computer systems. Studies have shown that productivity increases dramatically when visual user interfaces are provided. Known as GUIs (graphical user interfaces), each operating system provides its own variation. Some common graphical standards are Motif for UNIX systems and Microsoft Windows for PC-based systems.
- *Model Calculations*. Perform the calculations or algorithms (step-by-step procedures for solving a problem) intended by a program, e.g., process control, payroll calculations, or a Kalman filter.
- *Program Control.* Exerts control in the form of comparisons, branching, calling other programs, and iteration to carry out the logic of the program.
- *Message Processing.* There are several varieties of message processing. Help-message processing is the construct by which the program responds to requests for help from the end-user. Error-message processing is the automatic capability of the program to notify and then recover from an error during input, output, calculations, reporting, communications, etc. And, in object-oriented development environments, message processing implies the ability of program objects to pass information to other program objects.
- *Moving Data.* Programs store data in a data structure. Data can be moved between data structures within a program, moved from an external database or file to an internal data structure or from user input to a program's internal data structure. Alternatively, data can be moved from an internal data structure to a database or even to the user interface of an end-user. Sorting and formatting are data moving operations used to prepare the data for further operations.
- *Database.* A collection of data (objects[1]) or information about a subject or related subjects, or a system (for example, an engine in a truck or a personnel department in an organization). A database can include objects, such as forms and reports or a set of facts about the system (for example, the information in the personnel department needed about the employees in the company). A database is organized in such a way so as to be easily accessible to computer users. Its data is a representation of facts, concepts, or instructions in a manner suitable for processing by computers. It can be displayed, updated, queried, printed, and reports can be produced from it. A database can organize data in several ways including in a relational, hierarchical, network, or object-oriented format.
- *Data Declaration*. It describes data and data structures to a program. An example would be associating a particular data structure with its type (for example, data about a particular employee might be of type person).
- *Object.* A person, place, or thing, which could be physical or abstract. An object contains other more primitive objects (or data) and a set of operations to manipulate objects (or data). When brought to life, it knows things (called attributes) and can do things (to change itself or interact with other objects). For example, in a robotics system a robot object may contain the functions to move its own armature to the right, while it is coordinating with another robot to transfer yet another object. Objects can communicate with each other through a communications medium (e.g., message passing, radio waves, Internet).
- *Real-Time.* A software system that satisfies critical timing requirements. The correctness of the software depends on the results of computation, as well as on the time at which the results are produced. Real-time systems can have varied requirements such as performing a task within a specific deadline and processing data in connection with another process outside of the computer. Applications such as transaction processing, avionics, interactive office management, automobile systems, and video games are examples of real-time systems.

[1]Data and object are used interchangeably throughout this chapter to define information in a software program.

- *Distributed.* Any system in which a number of independent, interconnected processes can cooperate. The client/server model is one of the most popular forms of distribution in use today. In this model, a client initiates a distributed activity and a server carries out that activity.
- *Simulation.* The representation of selected characteristics of the behavior of one physical or abstract system by another system. For example, a software program can simulate an airplane or an organization or another software program.
- *Documentation.* It includes the description of requirements, specification, and design as well as written or generated documentation, which describe how each program within the larger system operates and can be used; and comments which describe the operation of the program that are stored internally in the program.
- *Tools.* The software programs used to design, develop, test, analyze, or maintain system designs or another software program and its documentation. They include code generators, compilers, editors, database management systems (DBMS), GUI builders, debuggers, operating systems, and software development and systems engineering tools referred to in the 1990s as computer-aided software engineering (CASE) tools and now often referred to as life-cycle design or life-cycle development environments, which combine a set of tools, including some of those listed above.

Although the reader should by now understand the dynamics of a line of source code, where that line of source code fits into the superorganism of software is dependent upon many variables. This includes the industry the reader hails from as well as the software development paradigm used by the organization.

As a base unit, a line of code can be joined with other lines of code to form many things. In a traditional software environment many lines of code form a program, sometimes referred to as an application program or just plain application. But lines of source code by themselves cannot be executed. First, source code must be run through what is called a compiler to create an object code. Next, the object code is run through a linker which is used to construct an executable code. Compilers are programs themselves. Their function is twofold. The compiler first checks the source code for obvious syntax errors and then, if it finds none, creates object code for a specific operating system. UNIX, Linux (a spinoff of UNIX), and NT are all examples of operating systems. An operating system can be thought of as a supervising program that controls the application programs that run under its control. Since operating systems (as well as computer architectures) can be different from each other, the object code resulting from the source code compiled for one operating system cannot be executed under a different kind of operating system—without a recompilation.

Solving a complex business or engineering problem often requires more than one program. One or more programs that run in tandem to solve a common problem is known collectively as a system. The more modern technique of object-oriented development dispenses with the notion of the program altogether and replaces it with a classification-oriented concept of an object.

Where a program can be considered a critical mass of code, which performs many functions in the attempt to solve a problem with little consideration for object boundaries, an object is associated with the code to solve a particular set of functions having to do with just that type of object. By combining objects, like molecules, it is possible to create more organized systems than those created by traditional means. Software development becomes a speedier and less error-prone process as well. Since objects can be reused, once tested and implemented, they can be placed in a library for other developers to reuse. The more objects in the library, the easier and quicker it is to develop new systems. And since the objects being reused have, in theory, already been warranted (i.e., they've been tested and made error-free), there is less possibility that object-oriented systems will have major defects.

The process of writing programs and/or objects is known as software development, or software engineering. It is composed of a series of steps or phases, collectively referred to as a development life cycle. The phases include (at a bare minimum) the following: an analysis or requirements phase, where the business problem is dissected and understood; a specification phase, where decisions are made as to how the requirements will be fulfilled (e.g., deciding what functions are allocated to software and what functions are allocated to hardware); a design phase, where everything from the GUI to the database to

the output is designed or selected as part of a design; an implementation or programming phase, where one or more tools are used to write and/or generate code; a testing (debugging) phase, where the code is tested against a business test case and errors in the program are found and corrected; an installation phase, where the systems are placed in production; and a maintenance phase, where modifications are made to the system. But different people develop systems in different ways. These different paradigms make up the opposing viewpoints of software engineering.

## 21.2 The Nature of Software Engineering

Engineers often use the term "systems engineering" to refer to the tasks of specifying, designing, and simulating a non-software system such as a bridge or electronic component. Although software may be used for simulation purposes, it is but one part of the systems engineering process. Software engineering, on the other hand, is concerned with the production of nothing but software.

In the 1970s industry pundits began to notice that the cost of producing large-scale systems was growing at a high rate and that many projects were failing or, at the very least, resulting in unreliable products. Dubbed the software crisis, its manifestations were legion and the most important include the following:

- *Programmer Productivity.* In government in the 1980s, an average developer using C was expected to produce 10 lines of code per day (an average developer within a commercial organization was expected to produce 30 lines a month); today the benchmark in the government is more like 2 to 5 lines a day while at the same time the need is dramatically higher than that, perhaps by several orders of magnitude, ending up with a huge backlog. Programmer productivity is dependent upon a plethora of vagaries—from expertise to complexity of the problem to be coded to the size of the program that is generated. The science of measuring the productivity of the software engineering process is called metrics. Just as there are many diverse paradigms in software engineering itself, there are many paradigms of software measurement. Today's metric formulas are complex and often take into consideration the following: cost, time to market, productivity on prior projects, data communications, distributed functions, performance, heavily used configuration, transaction rate, online data entry, end-user efficiency, online update, complex processing, reusability, installation ease, operational ease, and multiplicity of operational sites.
- *Defect Removal Costs*. The same variables that affect programmer productivity affect the cost of "debugging" the programs and/or objects generated by those programmers. It has been observed that the testing and correcting of programs consumes a large share of the overall effort.
- *Development Environment.* Development tools and development practices greatly affect the quantity and quality of software. Most of today's design and programming environments contain only a fragment of what is really needed to develop a complete system. Life-cycle development environments provide a good example of this phenomena. Most of these tools can be described either as addressing the upper part of the life cycle (i.e., they handle the analysis and design) or the lower part of the life cycle (i.e., they handle code generation). There are few integrated tools on the market (i.e., that seamlessly handle both upper and lower functionalities). There are even fewer tools that add simulation, testing, and cross-platform generation to the mix. Rare are the tools that seamlessly integrate system design to software development.
- *GUI Development.* Developing GUIs is a difficult and expensive process unless the proper tools are used. The movement of systems from a host-based environment to the workstation and/or PC saw the entry of countless GUI development programs onto the marketplace. But the vast majority of these GUI-based tools do not have the capability of developing the entire system (i.e., the processing component as opposed to merely the front-end). This leads to fragmented and error-prone systems. To be efficient, the GUI builder must be well integrated into the software development environment.

The result of these problems is that most of today's systems require more resources allocated to maintenance than to the original development of that system. Lientz and Swanson [4] demonstrate that the problem is, in fact, larger than the one originally discerned during the 1970s. Software development is

indeed complex, and the limitations on what can be produced by teams of software engineers given finite amounts of time, budgeted dollars, and talent have been amply documented by Jones [5].

Essentially the many paradigms of software engineering attempt to rectify the causes of declining productivity and quality. Unfortunately, this fails because current paradigms treat symptoms rather than the root problem. In fact, software engineering is itself extremely dependent upon both the software and hardware as well as the business environments upon which they sit [6].

SEI's process maturity grid very accurately pinpoints the root of most of our software development problems. The fact that a full 86% of organizations studied remain at the ad hoc or chaotic level indicate that only a few organizations (the remaining 14%) have adopted any formal process for software engineering. Simply put, 86% of all organizations react to a business problem by just writing codes. If they do employ a software engineering discipline, in all likelihood it is one that no longer fits the requirements of the ever-evolving business environment.

In the 1970s, the "structured methodology" was popularized. Although there were variations on the theme (i.e., different versions of the structured technique included the popular Gane–Sarson method and Yourdon method), for the most part, it provided a methodology to develop usable systems in an era of batch computing. In those days, online systems with even the dumbest of terminals were a radical concept and GUIs were as unthinkable as the fall of the Berlin Wall.

Although times have changed and today's hardware is one thousand times more powerful than when structured techniques were introduced, this technique still survives. And it survives in spite of the fact that the authors of these techniques have moved on to more adaptable paradigms, and more modern software development and systems engineering environments have entered the market.

In 1981, Finkelstein and Martin popularized "information engineering" [7] for the more commercially oriented users (i.e., those whose problems to be solved tended to be more database centered) which, to this day, is quite popular among mainframe developers with an investment in CASE strategies of the 1990s. Information engineering is essentially a refinement of the structured approach. However, instead of focusing on the data so preeminent in the structured approach, information engineering focuses on the information needs of the entire organization. Here business experts define high-level information models, as well as detailed data models. Ultimately, the system is designed from these models.

Both structured and information engineering methodologies have their roots in mainframe-oriented commercial applications. Today's migration to client/server technologies (where the organization's data can be spread across one or more geographically distributed servers while the end-user uses his or her GUI of choice to perform local processing), disables most of the utility of these methodologies. In fact, many issues now surfacing in more commercial applications are not unlike those that needed to be addressed earlier in the more engineering-oriented environments such as telecommunications and avionics.

Client/server environments are characterized by their diversity. One organization may store its data on multiple databases, program in several programming languages, and use more than one operating system, and hence, different GUIs. Since software development complexity is increased 100-fold in this new environment, a better methodology is required. Today's object-oriented techniques solve some of the problems. Given the complexity of the client/server environment, code trapped in programs is not flexible enough to meet the needs of this type of environment. We have already discussed how coding via objects rather than large programs engenders flexibility as well as productivity and quality through reusability. But object-oriented development is a double-edged sword.

While it is true that to master this technique is to provide dramatic increases in productivity, the sad fact of the matter is that object-oriented development, if done inappropriately, can cause problems far greater than problems generated from structured techniques. The reason for this is simple. The stakes are higher. Object-oriented environments are more complex than any other, the business problems chosen to be solved by object-oriented techniques are far more complex than other types of problems, and there are few if any conventional object-oriented methodologies and corollary tools to help the development team develop good systems. There are many flavors of object orientation. But with this diversity comes some very real risks. As a result, the following developmental issues must be considered before the computer is even turned on.

- Integration is a challenge and needs to be considered at the onset. With traditional systems, developers rely on mismatched modeling methods to capture aspects of even a single definition. Whether it be integration of object to object, module to module, phase to phase, or type of application to type of application, the process can be an arduous one. The mismatch of products used in design and development compounds the issue. Integration is usually left to the devices of myriad developers well into development. The resulting system is sometimes hard to understand and objects are difficult to trace. The biggest danger is there is little correspondence to the real world. Interfaces are often incompatible and errors usually propagate throughout development. As a result, systems defined in this manner can be ambiguous and just plain incorrect.
- Errors need to be minimized. Traditional methods including those that are object oriented can actually encourage the propagation of errors, such as propagating errors through the reuse of objects with embedded and inherited errors throughout the development process. Errors must be eliminated from the very onset of the development process before they take on a life of their own.
- Languages need to be more formal.[2] Although some languages are formal and others are friendly, it is hard to find languages both *formal* and *friendly*. Within environments where more informal approaches are used, lack of traceability and an overabundance of interface errors are a common occurrence. Recently, more modern software requirements languages have been introduced (for example, the Unified Modeling Language, UML [8]), most of which are informal (or semi-formal); some of these languages were created by "integrating" several languages into one. Unfortunately, the bad comes with the good—often, more of what is not needed and less of what is needed; and since the formal part is missing, common semantics need to exist to reconcile differences and eliminate redundancies.
- The syndrome of locked-in design needs to be eliminated. Often, developers are forced to develop in terms of an implementation technology that does not have an open architecture, such as a specific database schema or a GUI. Bad enough is to attempt an evolution of such a system; worse yet is to use parts of it as reusables for a system that does not rely on those technologies. Well thought-out and formal business practices and their implementation will help minimize this problem within an organization.
- Flexibility for change and handling the unpredictable must be dealt with up front. Too often it is forgotten that the building of an application must take into account its evolution. Users change their minds, software development environments change, and technologies change. Definitions of requirements in traditional development scenarios concentrate on the application needs of the user, but without consideration of the potential for the user's needs or environment to change. Porting to a new environment becomes a new development for each new architecture, operating system, database, graphics environment, or language. Because of this, critical functionality is often avoided for fear of the unknown, and maintenance, the most risky and expensive part of a system's life cycle, is left unaccounted for during development. To address these issues, tools and techniques must be used to allow cross technology and changing technology, as well as provide for changing and evolving architectures.
- Developers must prepare ahead of time for parallelism and distributed environments. Often, when it is known that a system is targeted for a distributed environment, it is first defined and developed for a single processor environment and then redeveloped for a distributed environment—an unproductive use of resources. Parallelism and distribution must be dealt with at the very start of the project.
- Resource allocation should be transparent to the user. Whether or not a system is allocated to distributed, asynchronous, or synchronous processors and whether or not two or ten processors are selected, with traditional methods, it is still up to the designer and developer to be concerned

[2] See Defining Terms for definition of formal.

with incorporating such detail into the application. There is no separation between the specification of what the system is to do vs. how the system does it. This results in far too much implementation detail to be included at the level of design. Once such a resource architecture becomes obsolete, it is necessary to redesign and redevelop those applications which have old designs embedded within them.

- Automation that minimizes manual work needs to replace "make work" automated solutions. In fact, automation itself is an inherently reusable process. If a system does not exist for reuse, it certainly does not exist for automation. But most of today's development process is needlessly manual. Today's systems are defined with insufficient intelligence for automated tools to use them as input. In fact, automated tools concentrate on supporting the manual process instead of doing the real work. Typically, developers receive definitions, which they manually turn into code. A process that could have been mechanized once for reuse is performed manually again and again. Under this scenario, even when automation attempts to do the real work, it is often incomplete across application domains or even within a domain, resulting in incomplete code such as shell code. The generated code is often inefficient or hardwired to a particular kind of algorithm, an architecture, a language, or even a version of a language. Often partial automations need to be integrated with incompatible partial automations or manual processes. Manual processes are needed to complete unfinished automations.
- Run-time performance analysis (decisions between algorithms or architectures) should be based on formal definitions. Conventional system definitions contain insufficient information about a system's run-time performance, including that concerning the decisions between algorithms or architectures. System definitions must consider how to separate the system from its target environment. Design decisions, where this separation is not taken into account, thus depend on analysis of results from ad hoc "trial and error" implementations and associated testing scenarios.
- The creation of reliable reusable definitions must be promoted, especially those that are inherently provided. Conventional requirements definitions lack the facilities to help find, create, use, and ensure commonality in systems. Modelers are forced to use informal and manual methods to find ways to divide a system into components natural for reuse. These components do not lend themselves to integration and, as a result, they tend to be error-prone. Because these systems are not portable or adaptable, there is little incentive for reuse. In conventional methodologies, redundancy becomes a way of doing business. Even when methods are object oriented, developers are often left to their own devices to explicitly make their applications object oriented. This is because these methods do not support all that is inherent to the process of object orientation.
- Design integrity is the first step to usable systems. Using traditional methods, it is not known if a design is a good one until its implementation has failed or succeeded. Usually, a system design is based on short-term considerations because knowledge is not reused from previous lessons learned. Development, ultimately, is driven towards failure. The solution is to have an inherent means to build reliable, reusable definitions.

Once these issues are addressed, software will cost less and take less time to develop. But time is of the essence. These issues are becoming compounded and even more critical as developers prepare for the distributed environments that go hand in hand with the increasing predominance of Internet applications.

With respect to the challenges described above, an organization has several options, ranging from one extreme to the other. The options include: (1) keep things the same; (2) add tools and techniques that support business as usual, but provide relief in selected areas; (3) bring in more modern but traditional tools and techniques to replace existing ones; (4) use a new paradigm with the most advanced tools and techniques that formalizes the process of software development, while at the same time capitalizing on software already developed; or (5) completely start over with a new paradigm that formalizes the process of software development and uses the most-advanced tools and techniques.

## 21.3 Development before the Fact

Thus far, this chapter has explained the derivation of software and attempted to show how it has evolved over time to become the true "brains" of any automated system. But, like a human brain, this software brain must be carefully architected to promote productivity, foster quality, and enforce control and reusability.

Traditional software engineering paradigms fail to see the software development process from the larger perspective of the superorganism described at the beginning of this chapter. It is only when we see the software development process as made of discrete, but well-integrated, components can we begin to develop a methodology that can produce the very benefits that have been promised by the advent of software decades ago.

Software engineering, from this perspective, consists of a methodology as well as a series of tools with which to implement the solution to the business problem at hand. But even before the first tool can be applied, the software engineering methodology must be deployed to assist in specifying the requirements of the problem. How can this be accomplished successfully in the face of the issues needed to be addressed outlined in the last section? How can this be accomplished in situations where organizations must develop systems that run across diverse and distributed hardware platforms, databases, programming languages, and GUIs when traditional methodologies make no provision for such diversity? And how can software be developed without having to fix or "cure" those myriad of problems, which result "after the fact" of that software's development?

What is required is a radical revision of the way we build software, an approach that understands how to build systems using the right techniques at the right time. First and foremost, it is a preventative approach. This means it provides a framework for doing things right the first time. Problems associated with traditional methods of design and development are prevented "before the fact" just by the way a system is defined. Such an approach would concentrate on preventing problems of development from even happening rather than letting them happen "after the fact," and fixing them after they have surfaced at the most inopportune and expensive point in time.

Consider such an approach in its application to a human system. To fill a tooth before it reaches the stage of a root canal is curative with respect to the cavity, but preventive with respect to the root canal. Preventing the cavity by proper diet prevents not only the root canal, but the cavity as well. To follow a cavity with a root canal is the most expensive alternative, to fill a cavity on time is the next most expensive, and to prevent these cavities in the first place is the least expensive option.

Preventiveness is a relative concept. For any given system, be it human or software, one goal is to prevent, to the greatest extent and as early as possible, anything that could go wrong in the life cycle process.

With a preventative philosophy, systems would be carefully constructed to minimize development problems from the very outset. A system could be developed with properties that controlled its very own design and development. One result would be reusable systems that promote automation. Each system definition would model both its application and its life cycle with built-in constraints—constraints that protect the developer, but yet do not take away his flexibility.

The philosophy behind preventative systems is that reliable systems are defined in terms of reliable systems. Only reliable systems are used as building blocks, and only reliable systems are used as mechanisms to integrate these building blocks to form a new system. The new system becomes reusable for building other systems.

Effective reuse is a preventative concept. That is, reusing something (e.g., requirements or code) that contains no errors to obtain a desired functionality avoids both the errors and the cost of developing a new system. It allows one to solve a given problem as early as possible, not at the last moment. But to make a system truly reusable, one must start not from the customary end of a life cycle, during the implementation or maintenance phase, but from the very beginning.

Preventative systems are the true realization of the entelechy construct where molecules of software naturally combine to form a whole much greater than the sum of its parts. Or one can think of constructing systems from the tinker toys of our youth. One recalls that the child never errs in building

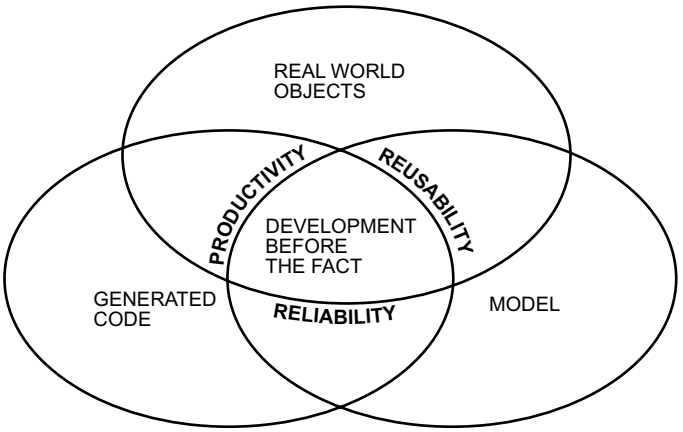



**FIGURE 21.1** The development before the fact paradigm.

magnificent structures from these tinker toys. Indeed, tinker toys are built from blocks that are architected to be perpetually reusable, perfectly integratable, and infinitely user-friendly.

One approach that follows this preventative philosophy is development before the fact (DBTF), as shown in Figure 21.1. Not yet in the mainstream, it has been used successfully by research and "trail blazer" organizations and is now being adopted for more commercial use. This technology is described in order to illustrate, by example, the potential that preventative approaches have.

Where traditional approaches begin the process of developing software after the fact, the DBTF paradigm is very much about beginnings. It was derived from the combination of steps taken to solve the problems of traditional systems engineering and software development. DBTF includes a technology, a language, and a process (or methodology) based on a formal theory.

## Language

Once understood, the characteristics of good design can be reused by incorporating them into a language for defining any system (i.e., not just a software system). One language based on DBTF is a formalism for representing the mathematics of systems. A system defined with this language has properties that come along "for the ride" that in essence control its own destiny. Based on a theory (DBTF) that extends traditional mathematics of systems with a unique concept of control, this formal, but friendly language has embodied within it a natural representation of the physics of time and space. With this language, every object is a system-oriented object (SOO), an integration that includes aspects of being function oriented (including dynamics) and object oriented. Instead of systems being object oriented, objects are systems oriented. All systems are objects and all objects are systems.

Because of this, many things heretofore not believed possible with traditional methods are possible. A DBTF system inherently integrates all of its own objects (and all aspects, relationships, and viewpoints of these objects) and the combinations of functionality, including timing, using these objects; maximizes its own reliability and flexibility to change (including the change of target requirements, static and dynamic architectures, and processes and as well reconfiguration in real time); capitalizes on its own parallelism and traceability; supports its own run-time performance analysis; and maximizes the potential for its own reuse (providing inherent resource allocation and reuse without need for the designer's intervention); and it provides the ability to automate design and development wherever and whenever possible. Each DBTF system is defined with built-in quality, built-in productivity, and built-in control.

The language—meta-language, really—is the key to DBTF. Its main attribute is to help the designer reduce the complexity and bring clarity into his thinking process, turning it into the ultimate reusable, which is wisdom itself. It can be used to define any aspect of any system and integrate it with any other aspect.

The crucial point is that these aspects are directly related to the real world and, therefore, the same language can be used to define system requirements, specifications, design, and detailed design for functional, resource, and resource allocation architectures throughout all levels and layers of seamless definition, including hardware, software, and peopleware.

This language based on DBTF can be used to define organizations of people, missile or banking systems, cognitive systems, as well as real-time or database environments and is, therefore, appropriate across industries, academia, or government.

## Technology

Real-world experience sets the stage for the DBTF technology. Having evolved over three decades, the theory has roots in the worlds of systems theory, formal methods, and object technology. The DBTF technology embodies the theory, the language supports its representation, and its automation supports its application and use. Each is evolutionary (in fact, recursively so), with experience feeding the theory and the theory feeding the language, which in turn feeds the automation. All are used, in concert, to design systems and build software.

The DBTF approach had its beginnings in 1968 with an empirical analysis of the Apollo space missions. A better way was needed to define and develop systems than the ones being used and available because the existing ones (just like the traditional ones today) did not solve the pressing problems. Research for developing software for man-rated missions led to the finding that interface errors accounted for approximately 75% of all errors found in the flight software during final testing (in traditional development, the figure is as high as 90%). Such errors include data flow, priority, and timing errors from the highest levels of a system to the lowest level of detail. Each error was categorized according to how it could be prevented just by the way a system is defined. This work led to a theory and methodology for defining a system that would eliminate all interface errors.

The first technology derived from this theory concentrated on defining and building reliable systems. Having realized the benefits of addressing one major issue, such as reliability, research continued to evolve by addressing other major issues the same way, that is, just by the way a system is defined [9–11].

DBTF is a function- and object-oriented approach based on a unique concept of control, which is lacking in any other software engineering paradigm. The foundations are based on a set of axioms and on the assumption of a universal set of objects. Each axiom defines a relation of immediate domination. The union of the relations defined by the axioms is control. Among other things, the axioms establish the relationships of an object for invocation, input and output, input and output access rights, error detection and recovery, and ordering during its developmental and operational states. Table 21.1 summarizes some of the properties of objects within DBTF systems.

## Process

Where software engineering fails is in its inability to grasp that not only the right paradigm (out of many paradigms) must be selected, but that the paradigm must be part of an environment that provides an integrated automated means to solve the problem at hand. What this means is that the paradigm must be coupled with an integrated system of tools with which to implement the results of utilizing that paradigm to develop the model of the system.

Essentially, the paradigm generates the model and a toolset must be provided to generate the system. DBTF provides this next-generation capability.

This DBTF approach is used throughout a life cycle, starting with requirements and continuing with functional analysis, simulation, specification, analysis, design, system architecture design, algorithm development, implementation, configuration management, testing, maintenance, and reverse engineering. Its users include end users, managers, system engineers, software engineers, and test engineers.

The DBTF process combines mathematical perfection with engineering precision. Its purpose is to facilitate the "doing things right in the first place" development style, avoiding the "fixing wrong things up" traditional approach. Its automation is developed with the following considerations: error prevention

**TABLE 21.1** System Oriented Object Properties of Development before the Fact

*Quality (better, faster, cheaper)*
- Reliable
- Affordable

*Reliable (better)*
- In control and under control
- Based on a set of axioms
  - –domain identification (intended, unintended)
  - –ordering (priority and timing)
  - –access rights: Incoming object (or relation), outgoing object (or relation)
  - –replacement
- Formal
  - –consistent, logically complete
  - –necessary and sufficient
  - –common semantic base
  - –unique state identification
- Error free (based on formal definition of "error")
  - –always gets the right answer at the right time and in the right place
  - –satisfies users and developers intent
- Handles the unpredictable
- Predictable

*Affordable (faster, cheaper)*
- Reusable
- Optimizes resources in operation and development
  - –in minimum time and space
  - –with best fit of objects to resources

*Reusable*
- *Understandable*, integratable and maintainable
- Flexible
- Follows standards
- Automation
- Common definitions
  - –natural modularity
    - -natural separation (e.g., functional architecture from its resource architectures);
    - -dumb modules
    - -an object is integrated with respect to structure, behavior and properties of control
  - –integration in terms of structure and behavior
  - –type of mechanisms
    - -function maps (relate an object's function to other functions)
    - -object type maps (relate objects to objects)
    - -structures of functions and types
  - –category
    - -relativity
      - instantiation
      - polymorphism
      - parent/child
      - being/doing
      - having/not having
    - -abstraction
      - encapsulation
      - replacement

*Handles the unpredictable*
- throughout development and operation
- Without affecting unintended areas
- Error detect and recover from the unexpected
- Interface with, change and reconfigure in asynchronous, distributed, real-time environment

*Flexible*
- Changeable without side effects
- Evolvable
- Durable
- Reliable
- Extensible
- Ability to break up and put together
  - –one object to many: modularity, decomposition, instantiation
  - –many objects to one: composition, applicative operators, integration, abstraction
- Portable
  - –secure
  - –diverse and changing layered developments
  - –open architecture (implementation, resource allocation, and execution independence)
  - –plug-in (or be plugged into) or reconfiguration of different modules
  - –adaptable for different organizations, applications, functionality, people, products

*Automation*
- the ultimate form of reusable
- formalize, mechanize, then automate
  - –it
  - –its development
  - –that which automates its development

*Understandable, integratable and maintainable*
- Reliable
- A measurable history
- Natural correspondence to real world
  - –persistence, create and delete
  - –appear and disappear
  - –accessibility
  - –reference
  - –assumes existence of objects
  - –real time and space constraints
  - –representation
  - –relativity, abstraction, derivation
- Provides user friendly definitions
  - –recognizes that one user's friendliness is another user's nightmare
  - –hides unnecessary detail (abstraction)
  - –variable, user selected syntax
  - –self teaching
  - –derived from a common semantic base
  - –common definition mechanisms
- Communicates with common semantics to all entities
- Defined to be simple as possible but not simpler

(*continued*)

**TABLE 21.1** System Oriented Object Properties of Development before the Fact (Continued)

| | |
|---|---|
| relation including function<br>typing including classification<br>form including both structure and behavior (for object types and functions)<br>-derivation<br>deduction<br>inference<br>inheritance | • Defined with integration of all of its objects (and all aspects of these objects)<br>• Traceability of behavior and structure and their changes (maintenance) throughout its birth, life and death<br>• Knows and able to reach the state of completion<br>–definition<br>–development of itself and that which develops it<br>-analysis<br>-design<br>-implementation<br>-instantiation<br>-testing<br>-maintenance |

*Note:* all underlined words point to a reusable.

*Source:* Hamilton, M., "Software Design and Development," *The Electronics Handbook,* CRC Press, Boca Raton, FL, 1996. With permission.

from the early stage of system definition, life cycle control of the system under development, and inherent reuse of highly reliable systems. The development life cycle is divided into a sequence of stages, including requirements and design modeling by formal specification and analysis, automatic code generation based on consistent and logically complete models, test and execution, and simulation.

The first step in building a DBTF system is to define a model with the language. This process could be in any phase of the developmental life cycle, including problem analysis, operational scenarios, and design. The model is automatically analyzed to ensure it was defined properly. This includes static analysis for preventive properties and dynamic analysis for user-intent properties.

In the next stage, the generic source code generator automatically generates a fully production-ready and fully integrated software implementation for any kind of application, consistent with the model, for a selected target environment in the language and architecture of choice. If the selected environment has already been configured, the generator selects that environment directly; otherwise, the generator is first configured for a new language and architecture.

Because of its open architecture, the generator can be configured to reside on any new architecture (or interface to any outside environment), e.g., to a language, communications package, an Internet interface, a database package, or an operating system of choice; or it can be configured to interface to the users own legacy code. Once configured for a new environment, an existing system can be automatically regenerated to reside on that new environment. This open architecture approach, which lends itself to true component-based development, provides more flexibility to the user when changing requirements or architectures, or when moving from an older technology to a newer one.

It then becomes possible to execute the resulting system. If it is software, the system can undergo testing for further user-intent errors. It becomes operational after testing. Application changes are always made to the requirements/specification definition—not to the code (the developer does not even need to change the code). Target architecture changes are made to the configuration of the generator environment (which generates one of a possible set of implementations from the model)—not to the code. If the real system is hardware or peopleware, the software system serves as a simulation upon which the real system can be based. Once a system has been developed, the system and the process used to develop it are analyzed to understand how to improve the next round of system development.

Seamless integration is provided throughout from systems to software, requirements to design to code to tests to other requirements and back again; level to level and layer to layer. The developer is able to trace from requirements to code and back again.

Given an automation that has these capabilities, it should be of no surprise that an automation of DBTF has been defined with itself and that it continues to automatically generate itself as it evolves with

**TABLE 21.2** A Comparison

| Traditional (After the Fact) | DBTF (Before the Fact) |
|---|---|
| *Interface errors* (*over 75% of all errors*) | *No interface errors* |
| Most found after implementation | All found before implementation |
| Some found manually | All found by automatic and static analysis |
| Some found by dynamic runs analysis | |
| Some never found | Always found |
| *Ambiguous requirements* | *Unambiguous requirements* |
| Informal or semiformal language | formal, but friendly language |
| Different phases, languages, and tools | All phases, same language and tools |
| Different language for other systems than for software | Same language for software, hardware and any other system |
| *Automation supports manual process* | *Automation does real work* |
| Mostly manual documentation, programming, test generation, traceability, etc. | Automatic documentation, programming, test generation, traceability, etc. |
| | 100% code automatically generated for any kind of software |
| *No guarantee of function integrity after implementation* | *Guarantee of function integrity after implementation* |
| *Systems not traceable or evolvable* | *Systems traceable and evolvable* |
| Locked in products, architectures, etc. | Open architecture |
| Painful transition from legacy | Smooth transition from legacy |
| Maintenance performed at code level | Maintenance performed at spec level |
| *Reuse not inherent* | *Inherent reuse* |
| Reuse is adhoc | Every object a candidate for reuse |
| Customization and reuse are mutually exclusive | Customization increases reuse pool |
| *Mismatched objects, phases, products, architectures and environment* | *Integrated & seamless objects, phases, products, architectures, and environment* |
| System not integrated with software | System integrated with software |
| Function oriented or object oriented | System oriented objects: integration of function, timing, and object oriented |
| GUI not integrated with application | GUI integrated with application |
| Simulation not integrated with software code | Simulation integrated with software code |
| *Automation not defined and developed with itself* | *Automation defined with and generated by itself* |
| | #1 in all evaluations |
| *Dollars wasted, error prone systems* | *Better, faster, cheaper systems* |
| Not cost-effective | 10 to 1, 20 to 1, 50 to 1…dollars saved |
| Difficult to meet schedules | Minimum time to complete |
| Less of what you need and more of what you don't need | No more, no less of what you need |

changing architectures and changing technologies. Table 21.2 contains a summary of some of the differences between the more modern preventative paradigm and the traditional approach.

A relatively small set of things is needed to master the concepts behind DBTF. Everything else can be derived, leading to powerful reuse capabilities for building systems. It quickly becomes clear why it is no longer necessary to add features to the language or changes to a developed application in an ad hoc fashion, since each new aspect is ultimately and inherently derived from its mathematical foundations.

## 21.4 Experience with DBTF

That preventative development is a superior alternative has been proven rather dramatically in several experiments. DBTF has been through many evaluations and competitions conducted and sponsored by leading academic institutions, government agencies, and commercial organizations. In every evaluation and competition this alternative came out on top. What set this alternative apart from the others was that it provided a totally integrated system design and development environment, whereas the traditional

methods resulted in an informal, difficult to integrate (including application modules as well as the products used to implement them), fragmented, more manual, and "after the fact" life-cycle process.

The National Test Bed of the U.S. Department of Defense sponsored an experiment in which it provided a development problem to each of three contractor/vendor teams chosen from a large pool of vendors and development environments, based upon a well-defined set of requirements. The application was a real-time, distributed, multiuser, client server system, which needed to be defined and developed under the government 2167A guidelines.

All teams were able to complete the first part, the definition of preliminary requirements. Two teams completed the detailed design. But only one team was able to generate complete, integrated, and fully production-ready code automatically; a major portion of this code was running in both C and Ada at the end of the experiment [12]. The team that was able to generate the production-ready code was using the 001 Tool Suite, a development environment based on the DBTF methodology.

## 21.5 Conclusion

Businesses that expected a big productivity payoff from investing in technology are, in many cases, still waiting to collect. A substantial part of the problem stems from the manner in which organizations are building their automated systems. While hardware capabilities have increased dramatically, organizations are still mired in the same old methodologies that saw the rise of the behemoth mainframes. Old methodologies simply cannot build the new systems.

There are other changes as well. Users demand much more functionality and flexibility in their systems. And given the nature of many of the problems to be solved by this new technology, these systems must also be error-free as well.

Where the biological superorganism has built-in control mechanisms fostering quality and productivity, until now the silicon superorganism has had none. Hence, the productivity paradox.

Often, the only way to solve major issues or to survive tough times is through nontraditional paths or innovation. One must create new methods or new environments for using new methods.

Innovation for success often starts with a look at mistakes from traditional systems. The first step is to recognize the true root problems, then categorize them according to how they might be prevented. Derivation of practical solutions is a logical next step. Iterations of the process entail looking for new problem areas in terms of the new solution environment and repeating the scenario. That is how DBTF came into being.

With DBTF all aspects of system design and development are integrated with one systems language and its associated automation. Reuse naturally takes place throughout the life cycle. Objects, no matter how complex, can be reused and integrated. Environment configurations for different kinds of architectures can be reused. A newly developed system can be safely reused to increase even further the productivity of the systems developed with it.

The paradigm shift occurs once a designer realizes that many of the old tools are no longer needed to design and develop a system. For example, with one formal semantic language to define and integrate all aspects of a system, diverse modeling languages (and methodologies for using them), each of which defines only part of a system, are no longer necessary. There is no longer a need to reconcile multiple techniques with semantics that interfere with each other.

DBTF can support a user in addressing many of the challenges presented in today's software development environments. There will, however, always be more to do to capitalize on this technology. That is part of what makes a technology like this so interesting to work with. Because it is based on a different premise or set of assumptions (set of axioms), a significant number of things can and will change because of it. There is the continuing opportunity for new research projects and new products. Some problems can be solved, because of the language, that could not be solved before. Software development as we know it will never be the same. Many things will no longer need to exist—they, in fact, will be rendered extinct, just as that phenomenon occurs with the process of natural selection in the biological system. Techniques for bridging the gap from one phase of the life cycle to another become obsolete. Testing procedures and tools

for finding most errors are no longer needed because those errors no longer exist. Tools to support programming as a manual process are no longer needed.

Compared to the development using traditional techniques, the productivity of DBTF developed systems has been shown to be significantly greater. Upon further analysis, it was discovered that the larger and more complex the system, the greater the productivity—the opposite of what one finds with traditional systems development. This is, in part, because of the high degree of DBTF's support of reuse. The larger a system, the more it has the opportunity to capitalize on reuse. As more reuse is employed, productivity continues to increase. Measuring productivity becomes a process of relativity—that is, relative to the last system developed.

Capitalizing on reusables within a DBTF environment is an ongoing area of research interest. An example is understanding the relationship between types of reusables and metrics. This takes into consideration that a reusable can be categorized in many ways. One is according to the manner in which its use saves time (which translates to how it impacts cost and schedules). More intelligent tradeoffs can then be made. The more we know about how some kinds of reusables are used, the more information we have to estimate costs for an overall system. Keep in mind also that the traditional methods for estimating time and costs for developing software are no longer valid for estimating systems developed with preventative techniques.

There are other reasons for this higher productivity as well, such as the savings realized and time saved due to tasks and processes that are no longer necessary with the use of this preventative approach. There is less to learn and less to do—less analysis, little or no implementation, less testing, less to manage, less to document, less to maintain, and less to integrate. This is because a major part of these areas has been automated or because of what inherently take place because of the nature of DBTF's formal systems language.

In the end, it is the combination of the technology and that which executes it that forms the foundation of successful software. Software is so ingrained in our society that its success or failure will dramatically influence both the operation and the success of an organization. For that reason, today's decisions about systems engineering and software development have far-reaching effects.

Software is a relatively young technological field that is still in a constant state of change. Changing from a traditional software environment to a preventative one is like going from the typewriter to the word processor. Whenever there is any major change, there is always the initial overhead needed for learning the new way of doing things. But, as with the word processor, progress begets progress.

Collective experience strongly confirms that quality and productivity increase with the increased use of properties of preventative systems. In contrast to the "better late than never" after the fact philosophy, the preventive philosophy behind DBTF is to solve—or if possible, prevent—a given problem as early as possible. Finding a problem statically is better than finding it dynamically. Preventing it by the way a system is defined is even better. Better yet is not having to define (and build) it at all.

Reusing a reliable system is better than reusing one that is not reliable. Automated reuse is better than manual reuse. Inherent reuse is better than automated reuse. Reuse that can evolve is better than one that cannot evolve. Best of all is reuse that ultimately approaches wisdom itself. Then, have the wisdom to use it.

The answer continues to be in the results just as in the biological system; and the goal is that the systems of tomorrow will inherit the best of the systems of today.

## References

1. Software Engineering Institute. *Capability Maturity Model,* Pittsburgh, PA: Carnegie, Mellon University, 1991.
2. Stroustrup, B., *The C++ Programming Language,* Reading, MA: Addison-Wesley, 1997.
3. Gosling, J., Joy, B., and Steele, G., *The Java Language Specification,* Reading, MA: Addison-Wesley, 1996.
4. Lientz, B.P., and Swanson, E.B., *Software Maintenance Management,* Reading, MA: Addison-Wesley, 1980.
5. Jones, T.C., *Program Quality and Programmer Productivity,* IBM Tech. Report TR02.764 January: 80, San Jose, CA: Santa Teresa Labs, 1977.

6. Keyes, J., *Handbook of E-Business,* Chapter F5, Hamilton, M., Defining e…com for e-Profits, RIA, 2000.
7. Martin, J., and Finkelstein, C.B., *Information Engineering,* Carnforth, Lancs, U.K.: Savant Institute, 1981.
8. Booch, G., Rumbaugh, J., and Jacobson, I., *The Unified Modeling Language User Guide,* Addison-Wesley, 1999.
9. Hamilton, M., "*Inside Development Before the Fact,*" Electronic Design, April 4, 1994, ES.
10. Hamilton, M., "*Development Before the Fact in Action,*" Electronic Design, June 13, 1994, ES.
11. Keyes, J., *The Ultimate Internet Developers Sourcebook,* AMACOM, to be published Fall 2001.
12. Software Engineering Tools Experiment-Final Report, Vol. 1, Experiment Summary, Table 1, Page 9, Department of Defense, Strategic Defense Initiative, Washington, D.C., 20301-7100, October 1992.

## Defining Terms

**Data Base Management System (DBMS):** The computer program that is used to control and provide rapid access to a database. A language is used with the DBMS to control the functions that a DBMS provides. For example, SQL is the language that is used to control all of the functions that a relational architecture-based DBMS provides for its users, including data definition, data retrieval, data manipulation, access control, data sharing, and data integrity.

**Formal:** A system defined in terms of a known set of axioms (or assumptions); it is, therefore, mathematically based (e.g., a DBTF system is based on a set of axioms of control). Some of its properties are that it is consistent and logically complete. A system is consistent if it can be shown that no assumption of the system contradicts any other assumption of that system. A system is logically complete if the assumptions of the method completely define a given set of properties. This assures that a model of the method has that set of properties. Other properties of the models defined with the method may not be provable from the method's assumptions. A logically complete system has a semantic basis (i.e., a way of expressing the meaning of that system's objects). In terms of the semantics of a DBTF system, this means it has no interface errors and is unambiguous, contains what is necessary and sufficient, and has a unique state identification.

**Graphical User Interface (GUI):** The ultimate user interface, by which the deployed system interfaces with the computer most productively, using visual means. Graphical user interfaces provide a series of intuitive, colorful, and graphical mechanisms that enable the end-user to view, update, and manipulate information.

**Interface:** A point of access in a boundary between objects or programs or systems. It is at this juncture that many errors surface. Software can interface with hardware, humans, and other software.

**Methodology:** A set of procedures, precepts, and constructs for the construction of software.

**Metrics:** A series of formulas that measure such things as quality and productivity.

**Software Architecture:** The structure and relationships among the components of software.

## Further Information

Hamilton, M. and Hackler, W. R., *Object Thinking: Development Before the Fact*, In Press.

Hamilton, M. and Hackler, W. R., Towards Cost Effective and Timely End-to-End Testing, HTI, prepared for Army Research Laboratory, Contract No. DAKF11-99-P-1236, July 17, 2000.

Keyes, J., *Internet Management,* Chapters 30–33, on 001-developed systems for the Internet, Auerbach, Boca Raton, FL, 2000.

Krut, Jr., B. "Integrating 001 Tool Support in the Feature-Oriented Domain Analysis Methodology" (CMU/SEI-93-TR-11, ESC-TR-93-188). Pittsburgh, PA: Software Engineering Institute, Carnegie-Mellon University, 1993.

McCauley, B. "Software Development Tools in the 1990s," AIS Security Technology for Space Operations Conference, July 1993, Houston, TX.

Ouyang, M. and Golay, M. W., "An Integrated Formal Approach for Developing High Quality Software of Safety-Critical Systems," Massachusetts Institute of Technology, Cambridge, MA, Report No. MIT-ANP-TR-035., September, 1995.