

MICROCONTROLLERS

Fundamentals and
Applications with PIC

Fernando E. Valdes-Perez
Ramon Pallas-Areny

 CRC Press
Taylor & Francis Group

Ltd.
016
312
f 25

MICROCONTROLLERS

Fundamentals and
Applications with PIC

MICROCONTROLLERS

Fundamentals and Applications with PIC

Fernando E. Valdes-Perez
Ramon Pallas-Areny



CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

Maxell, Ltd.
Ex. 2016
IPR2025-01312
3 of 25

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

© 2009 by Taylor & Francis Group, LLC
CRC Press is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works
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-13: 978-1-4200-7767-4 (Hardcover)

This book contains information obtained from authentic and highly regarded sources. Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access www.copyright.com (<http://www.copyright.com/>) or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

Trademark Notice: Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

Library of Congress Cataloging-in-Publication Data

Valdés Pérez, Fernando E.
Microcontrollers : fundamentals and applications with PIC / authors, Fernando
E. Valdes-Perez and Ramon Pallas-Areny.
p. cm.
Includes bibliographical references and index.
ISBN 978-1-4200-7767-4 (alk. paper)
1. Programmable controllers. 2. Microcontrollers. I. Pallàs-Areny, Ramón. II.
Title.

TJ223.P76V346 2009
629.8'9--dc22

2008044213

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

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

Maxell, Ltd.
Ex. 2016
IPR2025-01312
4 of 25

Contents

Preface.....	xi
The Authors	xiii
1 Introduction to Microcontrollers.....	1
1.1 Microprocessors and Microcontrollers: Characterization	1
1.2 Components of a Microcontroller.....	3
1.2.1 The Watchdog.....	5
1.2.2 Reset Signal.....	6
1.2.3 Low Consumption.....	7
1.2.4 Protection against Copying.....	8
1.3 Von Neumann and Harvard Architectures.....	9
1.4 CISC and RISC Architectures	11
1.5 Manufacturers of Microcontrollers and Microprocessors	12
2 PIC Microcontrollers.....	15
2.1 Main Characteristics of PIC Microcontrollers	15
2.1.1 The Arithmetic and Logic Unit (ALU) and the Working Register in PIC Microcontrollers.....	16
2.1.2 Machine Cycles and Execution of Instructions	17
2.1.3 Pipelining for Instruction Execution	18
2.1.4 Oscillators	19
2.1.5 Configuration Bits.....	21
2.1.6 Reset Options	22
2.1.7 Low-Power Consumption Mode.....	27
2.1.8 Watchdog Timer.....	27
2.2 PIC Microcontroller Families	28
2.2.1 Low-End Microcontrollers.....	29
2.2.2 Medium-End Microcontrollers	30
2.2.3 High-End Microcontrollers	32
3 Memory in Microcontrollers.....	39
3.1 Basic Concepts	39
3.1.1 Logic Organization of Memory	41
3.1.2 Types of Memory	43
3.2 Memory in Medium-End PIC Microcontrollers	44
3.2.1 Program Memory.....	44
3.2.1.1 Addressing Program Memory	45
3.2.1.2 Reading and Writing the Program Memory.....	47
3.2.2 RAM Data Memory	51

3.2.2.1	Addressing Data Memory.....	51
3.2.2.2	Special Function Registers (SFRs).....	54
3.2.3	EEPROM Data Memory	58
4	Instruction Set and Assembler Language Programming.....	61
4.1	Basic Concepts.....	61
4.1.1	Machine Code and Assembler Language	61
4.1.2	Structure of Instructions.....	64
4.1.3	Data Addressing Modes	65
4.1.4	The Stack	67
4.2	Instruction Set in Medium-End PIC Microcontrollers	69
4.2.1	Data Transfer Instructions.....	71
4.2.2	Arithmetic and Logic Instructions.....	72
4.2.3	Control Transfer Instructions.....	74
4.2.3.1	Unconditional Branches, Subroutine Calls, and Returns.....	74
4.2.3.2	Conditional Branches.....	78
4.2.4	Bit Manipulation Instructions.....	81
4.2.5	Other Instructions	81
4.3	Assembler Language Elements (for MPASM Assembler from Microchip)	82
4.3.1	Introduction.....	82
4.3.2	Expressions, Operations, and Operators	87
4.3.2.1	Arithmetic Operators.....	87
4.3.2.2	Logic and Boolean Operators	89
4.3.2.3	Logic Operators Using Direct Bit Manipulation	90
4.3.2.4	Assign Operators.....	90
4.3.2.5	Addressing Operators	92
4.3.3	Directives	93
4.3.3.1	General Use Directives	94
4.3.3.2	Directives for Relocatable Code	98
4.3.4	Macroinstructions.....	103
4.3.5	Organization of a Program in Assembler Language.....	105
4.4	Available Resources for Programming PIC Microcontrollers in Assembler Language.....	110
4.4.1	The MPASM Assembler	111
4.4.1.1	Absolute Code Generation.....	112
4.4.1.2	Relocatable Code Generation.....	112
4.4.1.3	Files Used and Generated during the Assembling Process	112
4.4.2	The Linker MPLINK	115
4.4.3	Library Manager MPLIB.....	117

5	Parallel Input and Output	121
5.1	Basic Concepts	121
5.1.1	Data Transfer Techniques	122
5.1.2	Input/Output Techniques.....	124
5.2	Parallel Ports in Medium-End PIC Microcontrollers	126
5.2.1	Port A.....	129
5.2.2	Port B.....	130
5.2.3	Port C	131
5.2.4	Ports D, E, F, and G	131
5.2.5	Parallel Slave Port (PSP)	132
5.3	Connection of Commonly Used Peripherals	134
5.3.1	Switches and LEDs	134
5.3.2	Matrix Keypads.....	138
5.3.3	Seven-Segment LEDs.....	145
5.3.4	Alphanumeric Liquid-Crystal Displays	148
6	Timers	157
6.1	Timers in PIC Microcontrollers.....	157
6.1.1	Timer0 Module.....	157
6.1.2	Timer1 Module.....	162
6.1.3	Timer2 Module.....	166
6.2	The CCP Module	168
6.2.1	Capture Mode.....	169
6.2.2	Compare Mode.....	174
6.2.3	PWM Mode.....	176
7	Interrupts	183
7.1	Basic Concepts	183
7.1.1	Interrupt Requests and Associated Resources	183
7.1.2	Servicing Interrupt Requests	185
7.1.3	Fixed and Vectored Interrupts	187
7.2	Interrupts in PIC Microcontrollers	189
7.2.1	Interrupt Sources and Associated Registers.....	189
7.2.2	Interrupt Service Subroutine Structure	194
7.3	Examples of Interrupt Applications	198
7.3.1	Real-Time Clock	198
7.3.2	Synchronization of Events to Real-Time Clock	202
7.3.3	Protection against Hardware Malfunctions	205
8	Serial Input and Output	207
8.1	Basic Concepts	207
8.1.1	Introduction to Serial Data Transmission	207
8.1.2	Asynchronous Communication	209
8.1.3	Synchronous Communication	209

8.1.4	Connection between Equipment: RS-232C Interface.....	210
8.1.5	The I ² C Bus.....	212
8.2	The USART Serial Port in PIC Microcontrollers.....	216
8.2.1	General Description.....	217
8.2.2	Asynchronous Mode.....	217
8.2.3	Synchronous Mode.....	220
8.2.4	Communication Speed.....	221
8.3	The Synchronous Serial Port in PIC Microcontrollers.....	223
8.3.1	SPI.....	223
8.3.2	I ² C Interface	228
9	Analog Input and Output: Signal Acquisition and Distribution	233
9.1	Structure of a System for Signal Acquisition and Distribution.....	233
9.1.1	Basic Functions of Measurement and Control Systems.....	233
9.1.2	Dynamic Range.....	236
9.1.3	Bandwidth.....	238
9.1.4	Signal Sampling	239
9.1.5	Architectures for Signal Acquisition: High-Level and Low-Level Multiplexing.....	240
9.2	The Front-End in Data Acquisition Systems	242
9.2.1	Attenuators	243
9.2.2	Amplifiers	247
9.2.3	Input Protections and Filters.....	251
9.2.4	Analog Multiplexers.....	253
9.2.5	Anti-Alias Filters.....	255
9.2.6	Sample-and-Hold Amplifier.....	257
9.2.7	A/D Converters.....	259
9.3	The 10-Bit A/D Converter Module in PIC Microcontrollers.....	262
9.3.1	Architecture of the Conversion Module.....	262
9.3.2	A/D Conversion Timing.....	266
9.3.3	A/D Conversion Module Programming.....	269
9.4	Calibration.....	271
9.5	Direct Sensor–Microcontroller Interface	273
9.6	Analog Back-End.....	276
9.6.1	D/A Converters	276
9.6.2	Analog Demultiplexing	277
9.6.3	Extrapolation Methods.....	277
9.6.4	PWM Outputs	278
9.6.5	Output Protections	280

Appendix: Acronyms 283
Bibliography..... 285
Index 287

The Authors

Fernando Eudaldo Valdés Pérez received his BS and MS degrees in electrical engineering from the Universidad de Oriente in Cuba in 1977 and 2001, respectively. He is an associate professor at the Center of Neuroscience Studies and Image and Signal Processing at the Universidad de Oriente. He has broad teaching experience, mostly focused on architecture programming of microprocessors, microcontrollers, and personal computers, as well as the statistical treatment of signals for biomedical applications. He is the main author of the textbook *Fundamentos Técnicos de Computación* (Fundamentals of Computing; ISPJAE, La Habana, 1986). His current research is focused on the acquisition and processing of cardiovascular signals. He has also worked on the design of hemodialysis monitoring systems.

Ramon Pallàs-Areny received the Ingeniero Industrial and Doctor Ingeniero Industrial degrees from the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 1975 and 1982, respectively. He is a professor of electronics engineering at the same university, and teaches courses in electronics and medical instrumentation. In 1989 and 1990 he was a visiting Fulbright Scholar, and in 1997 and 1998 he was an Honorary Fellow at the University of Wisconsin, Madison. His research includes instrumentation methods and sensors based on electrical impedance measurements, autonomous sensors, sensor interfaces, noninvasive physiological measurements and electromagnetic compatibility in electronic systems. He is the author of six books, the leading author of five books, and coauthor of two books on instrumentation in Spanish and Catalan. He is also coauthor, with John G. Webster, of *Sensors and Signal Conditioning*, 2nd edition (New York, Wiley, 2001), and *Analog Signal Processing* (New York, Wiley, 1999); and with Ferran Reverter on *Direct Sensor-to-Microcontroller Interface Circuits* (Barcelona, Marcombo, 2005). Dr. Pallàs-Areny was a recipient, with John G. Webster, of the 1991 Andrew R. Chi Prize Paper Award from the IEEE/Instrumentation and Measurement Society. In 2000 he received the Award for Quality in Teaching granted by the Board of Trustees of UPC, and in 2002 the Narcís Monturiol Medal from the Autonomous Government of Catalonia.

1

Introduction to Microcontrollers

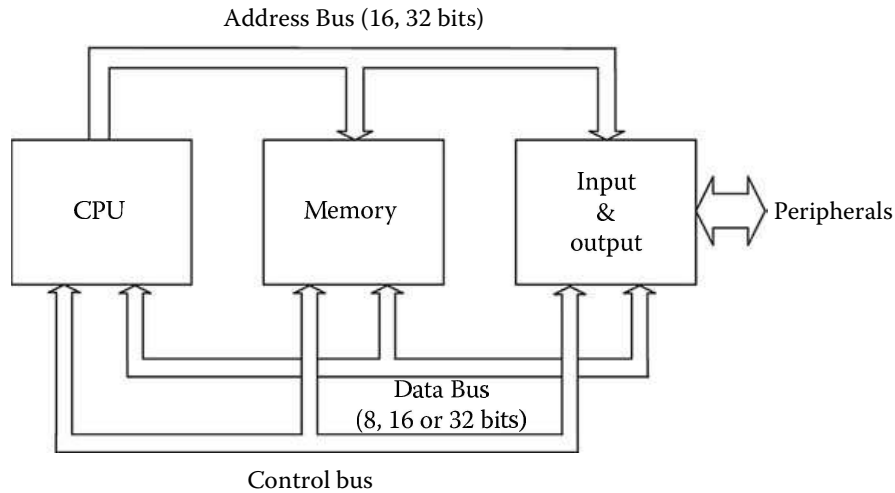
This chapter studies the structure and resources found in typical microcontrollers. It starts by introducing the concept of a microcontroller and exploring the differences between microcontrollers and microprocessors. It continues with the description of the resources that are available in microcontrollers, focusing again on how they differ from the resources available in microprocessors. The chapter then describes the von Neumann and Harvard architectures as well as how the reduced instruction set computer (RISC) and complex instruction set computer (CISC) architectures differ in their instruction sets. It finishes by describing the most common microcontrollers and listing their manufacturers.

1.1 Microprocessors and Microcontrollers: Characterization

Figure 1.1 shows the block diagram of a generic microcomputer. It consists of three fundamental blocks: central processing unit (CPU), memory, and input/output (I/O) system. These blocks are interconnected by groups of electrical lines called *buses*. The buses that transport memory or I/O addresses are called *address buses*; the buses that transport data or instructions are called *data buses*; and the buses that transport control signals are called *control buses*.

The CPU is the brain of the microcomputer, being under control of the program stored in memory. The tasks of the CPU are to fetch the instructions stored in memory, interpret those instructions, and execute them. The CPU also includes the circuitry necessary to perform arithmetic and logic operations with binary data. This special circuitry is called the arithmetic and logic unit (ALU).

In a microcomputer, the CPU is its *microprocessor*, which is the integrated circuit that carries out the operations described above. A microcontroller can be considered as a microcomputer built on a single integrated circuit or *chip*. Historically, microcontrollers appeared after microprocessors and followed independent paths. Microprocessors are mainly found in personal computers and workstations, as these require strong computational power, and the ability to manage large sets of data and instructions at a high speed. A very important parameter for microprocessors is the size of

**FIGURE 1.1**

Generic block diagram of a microcomputer. Here, the CPU is the microprocessor.

their internal registers (8, 16, 32, or 64 bits), as this determines the number of bits that can be processed simultaneously.

On the other hand, microcontrollers are used in a large variety of applications. They can be found in the automotive industry, communication systems, electronic instrumentation, hospital equipment, industrial equipment and applications, household appliances, toys, and so forth. Microcontrollers have been designed to be used in applications in which they have to carry out a small number of tasks at the lowest possible economic cost. They do this by executing a program permanently stored in their memory, whereas the input/output ports of the microcontroller are used to interact with the outside world. Therefore, the microcontroller becomes part of the application; it is a controller embedded in the system. Complex applications can use several microcontrollers, each one of them focusing on a small group of tasks.

The following generic requirements are important for microcontrollers and designs using microcontrollers:

1. Input/output resources. As opposed to microprocessors in which the emphasis is on computational power, microcontrollers put their emphasis on their input/output resources, such as the ability to handle interrupts, analog signals, number of different input and output lines, and so forth.
2. Optimization of space. It is important to use the smallest possible footprint at a reasonable cost. Given that the number of pins in a chip depends on its packaging, the footprint can be optimized by having one pin able to perform several different functions.

3. Using the most appropriate microcontroller for a given application. Microcontroller manufacturers have developed families of devices with the same instruction set but different hardware aspects, such as memory size, input/output devices, and so forth. This allows the designer to select the most appropriate device from a given family.
4. Protection against failure. It is critical for safety to guarantee that the microcontroller is executing the correct program. If for any reason the program goes astray, the situation has to be immediately corrected. Microcontrollers have a *watchdog timer* (WDT) to ensure that the program is being executed correctly. Watchdog timers do not exist in personal computers.
5. Low power consumption. Because batteries power many applications using microcontrollers, it is important to ensure the low power consumption of microcontrollers. Furthermore, the energy used when the microcontroller is not doing anything, for example, when it is waiting for an action from the user like a keyboard input, needs to be kept to a minimum. To do this, the microcontroller is set in sleeping state until it resumes the execution of the program.
6. Protection of programs against copies. The program stored in memory needs to be protected against unauthorized reading. To do this, the microcontrollers incorporate protection mechanisms against copying.

1.2 Components of a Microcontroller

Microcontrollers combine the fundamental resources available in a microcomputer such as the CPU, memory, and I/O resources in a single chip. Figure 1.2 shows the block diagram for a generic microcontroller.

Microcontrollers have an oscillator to generate the signal necessary to synchronize all internal operations. Although this can be a basic RC (resistance capacitor) oscillator, a quartz crystal (XTAL) is normally used due to its high frequency stability. The frequency of the oscillator has a direct influence on the speed at which program instructions are executed.

Similar to microcomputers, the CPU is the brain of the microcontroller. The CPU fetches the program instructions from their locations in memory one by one, interprets or decodes them, and executes them. The CPU also includes the ALU circuits for binary arithmetic and logic operations.

The microcontroller's CPU has different registers. Some of these registers are intended for general use, whereas others have a specific pur-

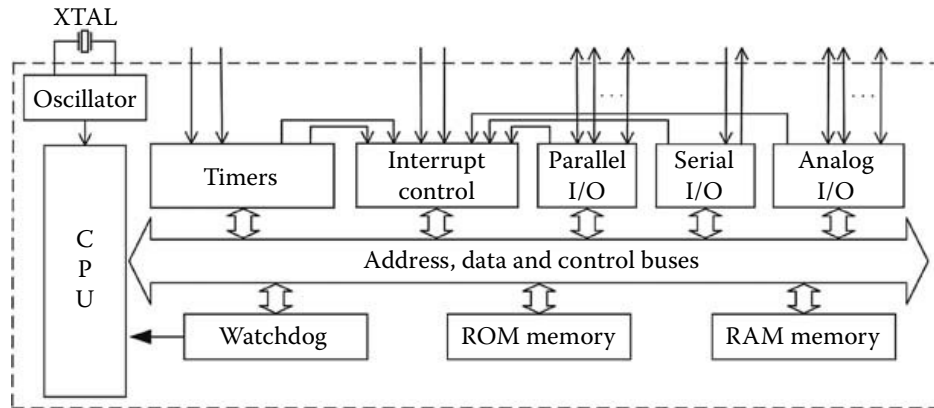


FIGURE 1.2

Basic block diagram of a microcontroller.

pose. Specific purpose registers include: instruction register, accumulator, status register, program counter, data address register, and stack pointer.

The *instruction register* (IR) stores the instruction that the CPU is executing. The programmer does not normally have access to the IR.

The *accumulator* (ACC) is a register associated with the arithmetic and logic operations that the ALU is carrying out. When executing any operation, one of the data needs to be in the ACC. The resulting value is also stored in the ACC. PIC microcontrollers do not have the ACC register. Instead, they have a working (W) register that is very similar to the ACC.

The *status register* (STATUS) contains the bits that show different characteristics related to the operations carried out by the ALU. These can be the sign of the resulting value (positive vs. negative), a flag to notify if the resulting value is zero, carry over, parity bits, and so forth.

The *program counter* (PC) is the CPU register where addresses of instructions are stored. Every time that the CPU looks for an instruction in the memory, the PC is increased, pointing to the following instruction. In an instant of time, the PC contains the address of the instruction that will be executed next. The control transfer instructions modify the value stored in the PC.

The *data address register* (DAR) stores data addresses from memory. This register plays a major role in indirect data addressing. Different types of microcontrollers use different specific names for the DAR. For example, PIC microcontrollers call this register the file select register (FSR).

The *stack pointer* (SP) stores data addresses in the stack. The stack and the SP register are studied in further detail in Chapter 4. PIC microcontrollers do not have an SP register.

The microcontroller memory stores both program instructions and data. Any microcontroller has two types of memory: random-access memory (RAM) and read-only memory (ROM). RAM can be read and written.

RAM is volatile memory, meaning that its data is lost when it is not powered. On the other hand, although ROM can only be read, it is non-volatile. The different types of technologies used for ROM such as EPROM (erasable programmable read-only memory), EEPROM (electrical erasable programmable read-only memory), OTP (one-time programmable), and FLASH are described in detail in Chapter 3. Both RAM and ROM are “random access” memories, meaning that the time to access specific data does not depend on its stored location. This is opposed to sequential access memories in which the time needed to access a specific memory cell depends on the location of the last accessed cell.

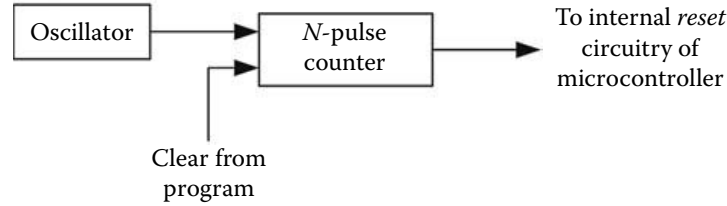
ROM is used to permanently store the program for the microcontroller, whereas RAM is used to temporarily store the data that will be manipulated by the program. An increasing number of microcontrollers use non-volatile memory such as EEPROM to store some of the data that is changed only sporadically. The size of ROM is larger than the size of RAM for two main reasons: First, most applications require programs that manipulate a relatively small number of data. Second, RAM has a larger footprint compared to ROM, and therefore it is more expensive than ROM.

Being the vehicle to communicate with the outside world, the I/O resources are very important in microcontrollers. I/O resources consist of the serial and parallel ports, timers, and interruption managers. Some microcontrollers also incorporate analog input and output lines associated with analog-to-digital (A/D) and digital-to-analog (D/A) converters. The resources needed to ensure the regular operation of the microcontrollers such as the watchdog are also considered part of the I/O resources.

Parallel ports are normally structured in groups of up to eight lines of digital inputs and outputs. It is normally possible to manipulate each one of these lines individually. Serial ports can be of different technologies such as RS-232C (Recommended Standard 232, Revision C), I²C (inter-integrated circuit), USB (universal serial bus), and Ethernet. In general, a microcontroller will have the largest possible number of I/O resources for the number of available pins in its integrated circuit package. To increase the performance, one physical pin can be connected to several internal blocks, and therefore that pin may carry out different functions depending on how the microcontroller has been configured.

1.2.1 The Watchdog

The watchdog timer (WDT) is a resource that can be found in most microcontrollers. As shown in Figure 1.3, the WDT consists of an oscillator and a binary counter of N bits. Although the oscillator can be the same oscillator used by the microcontroller, it is preferable to use an independent oscillator. The output of the counter is connected to the *reset input* for the microcontroller. The counting process can never be stopped, although the program being executed can periodically reset the counter to its initial value.

**FIGURE 1.3**

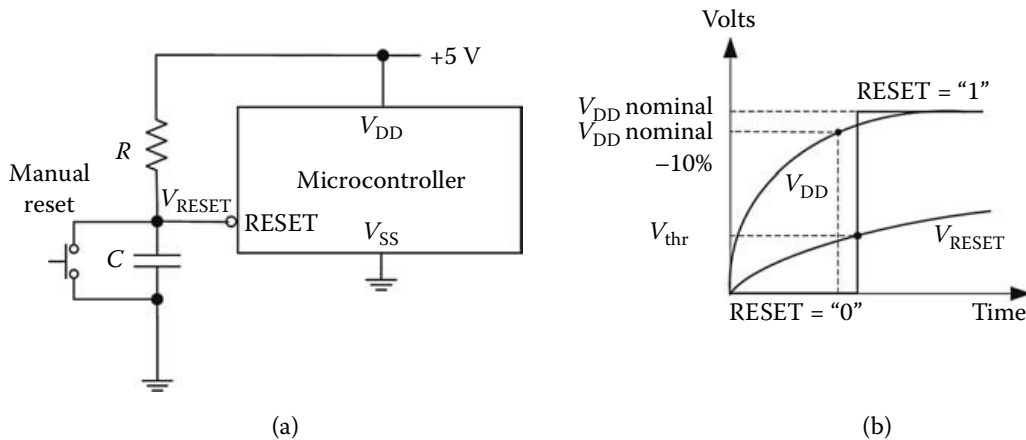
Basic watchdog diagram. Its output is connected to the internal circuitry to generate the reset signal.

Every pulse at the output of the oscillator becomes an input to the counter. When the counter reaches its maximum value, the output of the counter becomes active and gives a reset signal to the microcontroller. The goal of the designer is to avoid having the counter in the WDT reach its maximum value. Because, once started, the WDT cannot be stopped; the only way to avoid the reset signal is by setting the counter back to zero from the program that is being executed. This has to happen periodically and faster or the WDT counter will reach its maximum value. When the program is executed correctly, the WDT counter will never reach the maximum value. However, if the program becomes lost and stops executing the program, the WDT counter will reach its maximum value, will send the reset signal to the microcontroller, and the program will start executing from the beginning again. Therefore, the WDT is a critical element in a microcontroller, as it guarantees that the program will be executed continuously.

1.2.2 Reset Signal

Reset is an action that initializes microprocessors and microcontrollers. This initialization happens when a specific signal (called the *reset signal*) is applied to a specific pin (called the *reset pin*). The reset signal sets the program counter (PC) to a predetermined value, for example, $PC = 0$, making the microprocessor or microcontroller start executing the program commands from that specific memory address.

In a microcomputer, the reset signal can be applied manually (for example, when pressing a reset push button) or when the microcomputer boots up (*power-on reset*). Figure 1.4 shows the schematic of a circuit used to generate the reset signal either manually or through power-on. In the figure, V_{RESET} is the voltage applied to the reset pin, and V_{TH} is the threshold voltage for the pin. If $V_{\text{RESET}} < V_{\text{TH}}$, the device understands it as a logic value of 0 (RESET = 0). If $V_{\text{RESET}} > V_{\text{TH}}$, the device understands it as a logic value of 1 (RESET = 1). As shown in the schematic, the reset action occurs when RESET = 0.

**FIGURE 1.4**

Manual reset and power-on reset. (a) Typical reset circuit in a microcontroller. (b) Time evolution of voltages involved in reset signal.

The resistance (R) and the capacitor (C) make up a simple RC circuit with a time constant $\tau = RC$. In power-on, the voltage supply charges C through R . If τ is large enough, V_{RESET} is lower than V_{TH} during the time it takes for the voltage supply to become stable for the microcontroller to work correctly.

In addition to these two voluntary reset actions, the microcontroller may be unwillingly reset, for example, due to problems with its power supply (*power-glitch reset*, *brown-out reset*) or due to an action from the WDT. Power-glitch reset occurs when the applied voltage momentarily falls below a certain value, so the capacitor discharges to the point that $V_{\text{RESET}} < V_{\text{TH}}$. The reset originated by the WDT occurs when the WDT has not been refreshed and the output of the counter becomes active. This normally happens when the microcontroller has stopped executing the correct program in memory. It is very important for the microcontroller to generate a reset signal in this case to guarantee that the microcontroller will start executing instructions from a known memory address instead of reaching an unknown memory location that could damage the system. Some microcontrollers utilize specific bits in a register to signal that a reset action has taken place. This allows us to further investigate the reasons as to why the reset took place in order to take corrective actions.

1.2.3 Low Consumption

Because batteries power most applications using microcontrollers, power consumption has become a critical parameter. Power consumption in an integrated circuit depends on three factors: the technology used in the chip, the frequency of its oscillator, and the value of its voltage supply. CMOS (complementary metal-oxide semiconductor) is the preferred

technology for manufacturing microcontrollers due its low power needs. In static conditions only a very small leakage current flows through the gates. Its power consumption is only significant when switching logic states. Increasing the frequency of the oscillator increases the number of switching actions, and therefore its power consumption also increases. However, it is important to remember that in many applications the microcontroller is just waiting for an external event, such as a key being pressed, or an interrupt, before carrying out a task. Once finished, it returns to the waiting state. To further decrease its power consumption, it is a good idea to paralyze the microcontroller either totally or partially while it is waiting for an external event.

The best method to paralyze the microcontroller is to stop its main oscillator. This will force the main systems to be in a static mode waiting for an external action to start it again. When this happens, the microcontroller is said to be in *idle state*, *power down*, or *sleep mode*. Different microcontrollers have different methods to enter this low-power state. Some microcontrollers only need to modify a determined bit from a specific register, whereas other microcontrollers have a dedicated instruction for this purpose. The only way to leave this low-power mode is by means of an external interrupt or by a reset.

Example 1.1

8051 microcontrollers have two low-power modes: idle and power down. Any of these two modes can be entered by setting some specific bits of the power control (PCON) registry to 1. In idle mode, the CPU is paralyzed although the main oscillator and the other microcontroller blocks continue working. The microcontroller can leave this mode by means of an external interrupt or a reset. In power-down mode, the oscillator, and therefore the complete microcontroller, become paralyzed. It can only leave the power-down mode by means of a reset.

1.2.4 Protection against Copying

It is important to ensure the safety and protection of the information permanently stored in the microcontroller's memory and to avoid the program to be read or copied from memory once the device has been programmed.

Microcontrollers have resources to protect programs stored in their memory. This protection is normally optional; the programmer has to activate it. Some microcontrollers, like the programmable integrated circuit (PIC) family, can also be configured to prohibit reading of their memory once they have been programmed. Some other microcontrollers have open-memory architecture, that is, they allow the use of memory external to the device. In this case, the protection is done by encrypting the infor-

mation exchanged by the microcontroller and the external memory. This is typical for the 8051 family of microcontrollers.

Example 1.2

Program protection in 8051 microcontrollers. 8051 microcontrollers have open-memory architectures, allowing the use of external memory. These microcontrollers have two levels of program protection:

Level 1: The stored information is encrypted with an encryption word that can vary between 16 and 64 bits. The encryption is carried out using an XNOR operation between the encryption word and the program in memory. When the CPU reads the content in memory, it carries out another XNOR operation with one of the encryption bits, thus recovering the original bit. This makes it practically impossible to know the real information stored in memory if the encryption word is unknown.

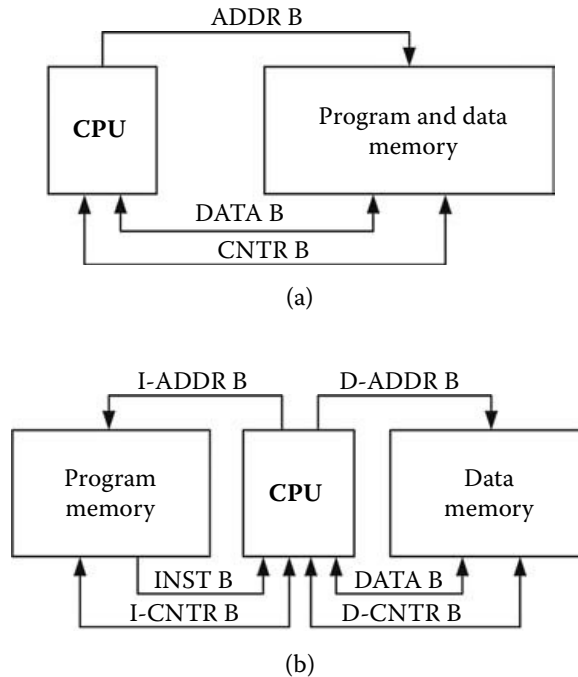
Level 2: A special registry in the microcontroller has security bits that can be programmed to limit total or partial access to the internal program memory.

1.3 Von Neumann and Harvard Architectures

The memory of a microcomputer, microprocessor, or microcontroller stores both data and instructions. Instructions need to move sequentially through the CPU to be decoded and executed. Data can be read from memory by the CPU or written in memory by the CPU. Therefore, the way that memory is organized and the way it communicates with the CPU determines the performance of the device. The two generic hardware models for memory structure are called Von Neumann and Harvard architectures.

Von Neumann architecture was proposed by the mathematician John von Neumann when he designed the *Electronic Numerical Integrator and Calculator* (ENIAC) at the University of Pennsylvania during World War II. He had the seminal idea of developing a stored-program computer. Harvard architecture was proposed by Howard Aiken when he developed the computers known as Mark I, II, III, and IV at Harvard University. These were the first computers to utilize different memories to store data and instructions separately, thus being a much different approach than the stored-program computer.

Figure 1.5 shows these two models. The von Neumann architecture uses a single memory to store instructions and data. This means that one unique address bus can access program instructions and data. Also, a unique data bus can transmit program instructions and data. The CPU

**FIGURE 1.5**

(a) von Neumann and (b) Harvard architectures. The von Neumann architecture uses a single memory connected to the CPU by using a single address bus (ADDR B), a single data bus (DATA B), and a single control bus (CNTR B). Harvard architecture uses different memories for data and instructions connected to the CPU by an instruction address bus (I-ADDR B), a data address bus (D-ADDR B), an instruction bus (INST B), a data bus (DATA B), an instruction control bus (I-CNTR B), and a data control bus (D-CNTR B).

sends the same control signal to read data or to read an instruction. There are no independent data or instructions control signals. Although ROM is used for instruction storage and RAM is used for data storage, the CPU is not concerned with this distinction and treats them the same way. From the CPU point of view, both ROM and RAM make up a single memory block to which the CPU sends control signals for addresses and data.

Harvard architecture uses different memories to store instructions and data. The program memory has its own address bus (instruction address bus), its own data bus (more properly called an instruction bus), and its own control bus. Data memory has its own address bus, data bus, and control bus independent from the instruction buses. The program memory can only be read when data memory can be read and written.

The von Neumann architecture uses fewer lines than the Harvard architecture, thus making a much simpler connection between CPU and memory. However, this structure does not allow simultaneous handling of data and instructions because there is only one bus. On the other hand, because it has different buses, Harvard architecture allows the handling of data and instructions simultaneously. This gives Harvard architecture an advantage in the speed of execution of programs.

In a microcomputer, the CPU is the microprocessor chip. Because it combines data and program in a single memory, a CPU implemented with the von Neumann architecture will need fewer pins and therefore will reduce the size of the CPU. For this reason, almost all microcomputers using a microprocessor have been developed using the von Neumann architecture. However, the situation is different in a microcontroller. In a microcontroller, the system components are located inside the same integrated chip and therefore there is no need to minimize pins. For this reason, Harvard architecture has been the chosen architecture for most microcontrollers, including the PIC family.

1.4 CISC and RISC Architectures

Complex set instruction computer (CISC) and *reduced instruction set computer (RISC)* are two different computer models classified according to their set of instructions. A CISC has a complex instruction set, whereas a RISC has a reduced instruction set.

When microprocessors and microcontrollers first appeared, the general trend was to give them the most powerful instruction set possible. Therefore, the CISC architecture became the prevalent mode. As time went on, the instructions increased in complexity to the point that the instruction set was a combination of very simple instructions (moving data from memory to the accumulator, for example), and very complex instructions, such as moving a chain of data between memory locations. The length of the instructions was different, the addressing mode became more complex, and in turn all this increased the complexity of the CPU and its size in the chip.

The CPU in RISC architectures has a short set of simple instructions. Each instruction carries out a very simple task (for example, moving data between CPU and memory), but it can be done very fast. Also, all the instructions have the same length. There are few addressing modes and all of them can be applied to any cell. This means that the CPU will be less complex, resulting in it being possible to increase the frequency of the oscillator in order to increase the speed at which operations are executed. Furthermore, as the CPU contains fewer transistors, they are less expensive to design and manufacture. CISC architecture has been the chosen mode for microprocessors and microcontrollers designed since the 1980s. PIC microcontrollers have RISC architecture.

1.5 Manufacturers of Microcontrollers and Microprocessors

Different microcontrollers that have the same core, that is, that share the same CPU and execute the same instruction set, are called a *family of microcontrollers*. Different devices within a family have the same core, but they differ in their I/O capabilities and their memory size. For example, all the microcontrollers in the 8051 family (MCS51) have a similar CPU and execute the same set of instructions. However, different family members have different numbers and types of I/O ports and also different memory types and sizes.

Microprocessors and microcontrollers are manufactured as stand-alone devices—chips that only contain the microprocessor or microcontroller. However, they can also be an embedded-processor core within a large density integration chip that the user will ultimately configure for a particular use. Programmable logic devices (PLD) such as field programmable gate arrays (FPGAs) are an example of such application. PLDs and FPGAs are large integration density circuits in which a user can select their function by choosing the appropriate interconnection elements. One of these elements may be the core of a microprocessor or a microcontroller that the user can connect to part of the memory and the chosen I/O devices. This allows the development of a custom microcontroller for a specific application, while having the advantage that this custom device is compatible with a standard device such as a PIC or 8051 as they both share the same core.

Several industries manufacture microcontrollers and microprocessors in any of the methods discussed earlier. The following is a list of microcontroller and microprocessor manufacturers, as well as of other devices that use a similar common core.

- Actel. FPGA with 8051 and ARM7 cores.
- Advanced Micro Devices (AMD). Microprocessors compatible with xx86.
- Altera. FPGA with Nios II core.
- Analog Devices. Architectures for digital signal processing based on 8052, ARM7, and other processors.
- Applied Micro Circuits Corp. (AMCC). Architectures based on the PowerPC microprocessor.
- ARC International. Architectures based on ARC 600, ARC 700, etc., microprocessors.
- ARM. Architectures based on ARM7, ARM9, ARM10, etc., microprocessor cores.
- Atmel. Architectures based on Marc 4, AVR, 8051, ARM7, ARM9, ARM11, PowerPC, and SPARC.

- Broadcom. Processors for communications and data networks with MIPS architecture.
- Cambridge Consultants. Architectures based on XAP1, XAP2, and XAP3 core processors.
- Cavium Networks. Architectures based on MIPS.
- Cirrus Logic. Architectures based on ARM.
- Cradle Technologies. Digital signal processors: CT3400 and CT3600.
- Cyan Technology. Microcontroller eCOG1k.
- Cybernetic Micro Systems. ASICs with microcontroller P-51.
- Cypress Microsystems. Devices with PSoC (Programmable System-on-Chip) architecture.
- Dallas Semiconductor. 8051-compatible microcontrollers.
- EM Microelectronics. Very low consumption EM6812.
- Freescale Semiconductor (from Motorola). Microcontrollers 68HC05, 68HC08, 68HC11, 68HC12, and 68HC16. DSPs. Processors ColdFire and PowerQuicc with PowerPC core.
- Fujitsu Microelectronics America. Microcontrollers FR80, MB9140x, and F2MC-8FX.
- Goal Semiconductor. Architectures based on 8051.
- Holtek Semiconductor. Microcontroller HT8.
- Hyperstone. Digital Signal Processors E1-32XSR/XSRU, HyNet32S, etc.
- Infineon Technologies (formerly Siemens). Microcontrollers C500, C800, C166, TriCore, etc.
- Infrant Technologies. Microcontrollers for data networks.
- Integrated Device Technology (IDT). Data Communications processors based on MIPS architecture.
- Intel. Microcontrollers from families MCS51, MCS151, MCS251, MCS96, MCS296, etc. Microprocessors xx86, IXP4xx, etc.
- Microchip Technology. Microcontrollers PIC (PICmicro) and digital signal controllers dsPIC.
- MIPS Technologies. Processors MIPS (Microprocessor without Interlocked Pipeline Stages).
- National Semiconductor. Microcontrollers COP8 and CR16, and microprocessors NS32000.
- NEC Electronics America. Microcontrollers 78K0, V850, and others.
- NetSilicon. Processors based on ARM7 and ARM9 cores.
- NXP Semiconductors (formerly Philips Semiconductors). Microcontrollers with 8051, ARM7, and ARM9 cores.

- Oki Semiconductor. Microcontrollers with ARM core.
- PMC-Sierra. MIPS-based processors.
- Rabbit Semiconductor. Processors Rabbit 2000 and 3000.
- Renesas Technology (formerly Hitachi). Microcontrollers R8, H8, and others.
- Sharp Microelectronics. Microcontrollers BlueStreak with ARM7 and ARM9 core.
- Silicon Laboratories. Microcontrollers with 8051 core.
- Silicon Storage Technology. Microcontrollers with 8051 core.
- STMicroelectronics. Microcontrollers with 8051 and ARM7 cores.
- Texas Instruments (TI). Digital signal processors TMS370 and TMS470. Microcontrollers MSP430.
- Toshiba America Electronic Components. Microcontrollers CISC and RISC.
- Ubicom. Microcontrollers SX, IP2000, and IP3000.
- Xemics. Microcontrollers with CoolRISC core.
- Xilinx. FPGA with PowerPC cores.
- ZiLOG. 8-bit microcontrollers with Z8 and Z80 architectures.

MICROCONTROLLERS

Fundamentals and Applications with PIC

Microcontrollers exist in a wide variety of models with varying structures and numerous application opportunities. Despite this diversity, it is possible to find consistencies in the architecture of most microcontrollers. **Microcontrollers: Fundamentals and Applications with PIC** focuses on these common elements to describe the fundamentals of microcontroller design and programming. Using clear, concise language and a top-to-bottom approach, the book describes the parts that make up a microcontroller, how they work, and how they interact with each other. It also explains how to program medium-end PICs using assembler language.

FEATURES

- Offers a complete and well-organized resource for those interested in developing designs with PIC microcontrollers
- Focuses on current topics of interest to electronic designers and experimenters
- Provides clarity for those just beginning to experiment with microcontrollers, as well as the depth of information needed for more advanced users
- Includes aspects of analog signals, including acquisition and processing of external signals
- Describes uses for a wide variety of applications

Using practical examples and applications to supplement each topic, this volume provides the tools to thoroughly grasp the architecture and programming of microcontrollers. The book avoids overly specific details so readers are quickly led toward design implementation. After mastering the material in this text, they will understand how to efficiently use PIC microcontrollers in a design process.



CRC Press
Taylor & Francis Group
an **informa** business

6000 Broken Sound Parkway, NW
Suite 300, Boca Raton, FL 33487
270 Madison Avenue
New York, NY 10016
2 Park Square, Milton Park
Abingdon, Oxon OX14 4RN, UK

77678

ISBN: 978-1-4200-7767-4



9 781420 077674

www.crcpress.com

Maxell, Ltd.

Ex. 2016

IPR2025-01312

25 of 25