BRIEF CONTENTS

CHAPTERS

APPENDICES

See the following websites to download the chapters which are labeled as "*(Web)*":

http://www.MicroDigitalEd.com

http://www.NicerLand.com

CONTENTS

PREFACE

The AVR is a widely used microcontroller. This book is intended for use in college-level courses teaching microcontrollers and embedded systems. It not only establishes a foundation of Assembly language programming, but also provides a comprehensive treatment of AVR interfacing for engineering students. From this background, the design and interfacing of microcontroller-based embedded systems can be explored. This book can also be used by practicing technicians, hardware engineers, computer scientists, and hobbyists.

Prerequisites

Readers should have had an introductory digital course. For the AVR C programming sections of the book, a basic knowledge of C programming is required.

Overview

A systematic, step-by-step approach is used to cover various aspects of AVR C and Assembly language programming and interfacing. Many examples and sample programs are given to clarify the concepts and provide students with an opportunity to learn by doing. Review questions are provided at the end of each section to reinforce the main points of the section.

Chapter 0 covers number systems (binary, decimal, and hex), and provides an introduction to basic logic gates and computer memory. This chapter is designed especially for students, such as mechanical engineering students, who have not taken a digital logic course or those who need to refresh their memory on these topics.

Chapter 1 discusses the history of the AVR and features of the members such as ATmega328. It also provides a list of various members of the AVR family.

Chapter 2 discusses the internal architecture of the AVR and explains the use of a AVR assembler to create ready-to-run programs. It also explores the program counter and the flag register.

In Chapter 3 the topics of loop, jump, and call instructions are discussed, with many programming examples.

Chapter 4 is dedicated to the discussion of I/O ports. This allows students who are working on a project to start experimenting with AVR I/O interfacing and start the hardware project as soon as possible.

Chapter 5 is dedicated to arithmetic, logic instructions, and programs.

Chapter 6 covers the AVR advanced addressing modes and explains how to access the data stored in the look-up table, as well as how to use EEPROM to store data and how to do macros.

The C programming of the AVR is covered in Chapter 7. We use the Atmel Studio compiler in this chapter and throughout the book.

In Chapter 8 we discuss the hardware connection of the AVR chip.

Chapter 9 describes the AVR timers and how to use them as event counters.

Chapter 10 provides a detailed discussion of AVR interrupts with many examples on how to write interrupt handler programs.

Chapter 11 is dedicated to serial data communication of the AVR and its interfacing to the RS232. It also shows AVR communication with COM ports of the x86 PC and compatible computers.

Chapter 12 shows AVR interfacing with LCD and keyboard.

Chapter 13 shows AVR interfacing with real-world devices such as DAC chips, ADC chips, and sensors.

Chapter 14 covers the basic interfacing of the AVR chip to relays, optoisolators, and stepper motors.

In Chapter 15 we cover how to use AVR timers to generate waves and explain how to capture waves to measure period and duty cycle.

Chapter 16 shows PWM and basic interfacing to DC motors.

Chapter 17 covers the SPI bus protocol and describes how to interface 7-segment displays using MAX7221.

Finally, Chapter 18 shows how to connect and program the DS1307 real-time clock chip using the TWI (I2C) bus protocol.

The appendices have been designed to provide some useful reference material required for the topics covered in the book. Appendix A provides the table of ASCII characters. Appendix B describes each AVR instruction in detail, with examples. Appendix B also provides the clock count for instructions. Appendix C examines IC interfacing and logic families, as well as AVR I/O port interfacing and fan-out. Make sure you study this section before connecting the AVR to an external device. In Appendix D, the use of flowcharts and pseudocode is explored. Appendix E describes the basics of wire wrapping. Appendix F discussed the fuse bits. Appendix G is for students familiar with 8051 architectures who need to make a rapid transition to AVR architecture.

Lab Manual, PowerPoint® Slides, and other support materials

The lab manual covers some very basic labs and can be found at the **www.MicroDigitalEd.com** website. The more advanced and rigorous lab assignments are left up to the instructors depending on the course objectives, class level, and whether the course is graduate or undergraduate. The support materials for this text and other books by the authors can be found on this website, too.

Compilers

We use the Atmel Studio compiler IDE from Atmel throughout the book. The Atmel Studio compiler is available for free from the Atmel website:

http://www.Atmel.com
http://www.microchip.com

Trainer board

You can use Arduino UNO as the trainerboard for this book. It is a very low priced trainer board and is available all arround the world.

The tutorials for the compiler and Trainer board can be found on the following websites:

http://www.MicroDigitalEd.com
http://www.NicerLand.com

CHAPTER 1

THE AVR MICROCONTROLLER: HISTORY AND FEATURES

OBJECTIVES

Upon completion of this chapter, you will be able to:

- **>> Compare and contrast microprocessors and microcontrollers**
- **>> Describe the advantages of microcontrollers for some applications**
- **>> Explain the concept of embedded systems**
- **>> Discuss criteria for considering a microcontroller**
- **>> Explain the variations of speed, packaging, memory, and cost per unit and how these affect choosing a microcontroller**
- **>> Compare and contrast the various members of the AVR family**
- **>> Compare the AVR with microcontrollers offered by other manufacturers**

This chapter begins with a discussion of the role and importance of microcontrollers in everyday life. In Section 1.1 we also discuss criteria to consider in choosing a microcontroller. In addition, we provide a brief discussion of alternatives to the AVR chip such as the PIC and ARM microcontrollers. Section 1.2 covers various members of the AVR family and their features.

SECTION 1.1: MICROCONTROLLERS

The evolution of Microprocessors and Microcontrollers

In early computers, CPUs were designed using a number of vacuum tubes. The vacuum tube was bulky and consumed a lot of electricity. The invention of transistors, followed by the IC (Integrated Circuit), provided the means to put a CPU on printed circuit boards. The advances in IC technology allowed putting the entire CPU on a single IC chip. This IC was called a *microprocessor*. Some of the microprocessors are the x86 family of Intel used widely in desktop computers, and the 68000 of Motorola. The microprocessors do not contain RAM, ROM, or I/O peripherals. As a result, they must be connected externally to RAM, ROM and I/O, as shown in Figure 1-1A.

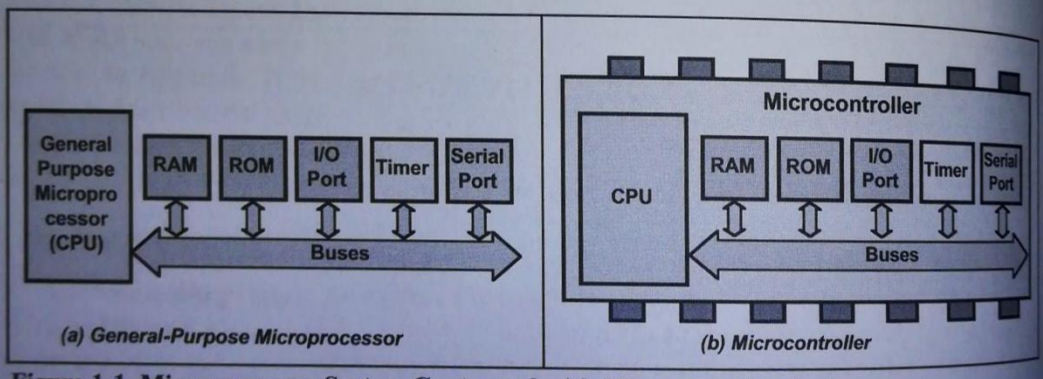

Figure 1-1. Microprocessor System Contrasted with Microcontroller System

In the next step, the different parts of a system, including CPU, RAM, ROM, and I/Os, were put together on a single IC chip and it was called *microcontroller* or *MCU* (Micro Controller Unit). Figure 1-1B shows the simplified view of the internal parts of microcontrollers.

Types of Computers

Typically, computers are categorized into 3 groups: desktop computers, servers, and embedded systems.

Desktop computers, including PCs, tablets, and laptops, are general purpose computers. They can be used to play games, read and edit articles, and do any other task just by running the proper application programs. The desktop computers use microprocessors.

In contrast, embedded systems are special-purpose computers. In embedded system devices, the software application and hardware are embedded together and are designed to do a specific task. A printer is an example of an embedded system because the processor inside it performs one task only; namely, getting the

Home
Appliances
Intercom
Telephones
Security systems
Garage door openers
Answering machines
Fax machines
Home computers
TVs
Cable TV tuner
VCR
Camcorder
Remote controls
Video games
Cellular phones
Musical instruments
Sewing machines
Lighting control
Paging
Camera
Pinball machines
Toys
Exercise equipment
Office
Telephones
Computers
Security systems
Fax machine
Microwave
Copier
Laser printer
Color printer
Paging
Auto
Trip computer
Engine control
Air bag
ABS
Instrumentation
Security system
Transmission control
Entertainment
Climate control
Cellular phone
Keyless entry

Table 1-1: Some Embedded Products Using Microcontrollers

data and printing it. In most cases embedded systems run a fixed program and contain a microcontroller. It is interesting to note that embedded systems are the largest class of computers though they are not normally considered as computers by the general public. Table 1-1 lists some embedded products.

Servers are the fast computers which might be used as web hosts, database servers, and in any application in which we need to process a huge amount of data such as weather forecasting. Similar to desktop computers, servers are made of microprocessors but, multiple processors are usually used in each server. Both servers and desktop computers are connected to a number of embedded system devices such as mouse, keyboard, disk controller, Flash stick memory and so on.

Designing embedded systems using MCUs, microprocessors and SoCs

In many applications, the space used, the power consumed, and the price per unit are critical considerations. These applications most often require some I/O operations to read signals and turn on and off certain devices. Since the microcontrollers are cheap and small, and have low power consumption they are the preferred choice for many embedded systems and they are widely used in embedded system products.

But sometimes microcontrollers are inadequate for a task. For this reason, sometimes general-purpose microprocessors are used to design embedded systems. in recent years many manufacturers of general-purpose microprocessors such as Intel, Freescale Semiconductor (formerly Motorola), and AMD (Advanced Micro Devices, Inc.) have targeted their microprocessors for the high end of the embedded market. Intel and AMD push their x86 processors for both the embedded and desktop PC markets. Currently, because of Linux and Windows standardization, in these embedded systems Linux and Windows operating systems are widely used. In many cases, using the operating systems shortens development time because a vast library of software already exists for the Linux and Windows platforms. The fact that Windows and Linux are widely used and well-understood platforms means that developing a Windows-based or Linux-based embedded product reduces the cost and shortens the development time considerably.

Sometimes the available microprocessors and microcontrollers cannot fullfill the needs of a design. (e.g. a small medical robat which should move in vessels). In such cases, it is common to integrate the most parts of the system into a single chip. Such a chip is called an *SoC (System on Chip)*. In recent years, companies have begun to sell *Field-Programmable Gate Array* (FPGA) and *Application-Specific Integrated Circuit* (ASIC) libraries for their microcontrollers. This makes the production of the new chips easier.

A Brief History of Microcontrollers

In the 1980s and 1990s, Intel and Motorola dominated the field of microprocessors and microcontrollers. Intel had the x86 (8088/86, 80286, 80386, 80486, and Pentium). Motorola (now NXP/Freescale) had the 68xxx (68000, 68010, 68020, etc.). Many embedded systems used Intel's 32-bit chips of x86 (386, 486, Pentium) and Motorola's 32-bit 68xxx for high-end embedded products such as routers. For example, Cisco routers used 68xxx for the CPU. At the low end, the 8051 from Intel and 68HC11 from Motorola were the dominant 8-bit microcontrollers. With the introduction of PIC from Microchip and AVR from Atmel (now Microchip), they became major players in the 8-bit market for microcontroller. At the time of this writing, AVR and PIC are the leaders in terms of volume for 8-bit microcontrollers. In the late 1990s, the ARM microcontroller started to challenge the dominance of Intel and Motorola in the 32-bit market. Although both Intel and Motorola used RISC features to enhance the performance of their microprocessors, due to the need to maintain compatibility with legacy software, they could not make a clean break and start over. Intel used massive amounts of gates to keep up the performance of x86 architecture and that in turn increased the power consumption of the x86 to a level unacceptable for battery-powered embedded products. Meanwhile Freescale (Motorola) streamlined the instructions of the 68xxx CPU and created a new line of microprocessors called ColdFire, while at the same time worked with IBM to design a new RISC processor called PowerPC. While both PowerPC and Coldfire are still alive and being used in the 32-bit market, it is ARM which has become the leading microcontroller in the 32-bit market.

Currently Available Microcontrollers

There are many microcontrollers available in the market. Some of them are listed in Tables 1-2A and 1-2B.

Choosing a microcontroller

Each of the microcontrollers has a unique instruction set and register set; therefore, they are not compatible with each other. Programs written for one will not run on the others. With all these different microcontrollers, what criteria do designers consider in choosing one? Three criteria in choosing microcontrollers are as follows:

1. **Chip characteristics:** The first and foremost criterion in choosing a microcontroller is that it must meet the task at hand efficiently and cost effectively. Some of the factors are:
   (a) Speed: What is the highest speed that the microcontroller supports?
   (b) Packaging: Does it come in a DIP (dual inline package) or a QFP (quad flat package), or some other packaging format? This is important in terms of space, assembling, and prototyping the end product.
   (c) Power consumption: This is especially critical for battery-powered products.
   (d) The amount of RAM and ROM on the chip.
   (e) Peripherals.
   (f) The number of I/O pins and the timer on the chip.
   (g) Ease of upgrade to higher-performance or lower-power-consumption ver-

Table 1-2A: Some Microcontrollers

8-bit

- 8051
- AVR (Microchip, formerly Atmel)
- PIC16, PIC18 (Microchip)
- S08 or 64HCS08 (NXP, formerly Freescale)

16-bit

- PIC24, dsPIC (Microchip)
- HCS12 (NXP/Freescale)
- MSP430 (TI)

32-bit

- ARM
- AVR32 (Atmel)
- ColdFire (Freescale)
- MIPS32
- PIC32 (Microchip)
- PowerPC
- SuperH
- TriCore (Infineon)

Table 1-2B: Some of the Companies that Produce Widely Used 8-bit Microcontrollers

Company	Website	Architecture
Atmel (now Microchip)	http://www.atmel.com	AVR and 8051
Microchip	http://www.microchip.com	PIC16, PIC18, AVR, 8051
Intel	http://www.intel.com	8051
Zilog	http://www.zilog.com	Z8 and Z80
Dallas Semi/Maxim	http://www.maxim-ic.com	8051
NXP (formerly philips)	http://www.nxp.com	68HC11/HCS08/8051

sions.

(h) Price: This is important in terms of the final cost of the product in which a microcontroller is used.

2. **Ease of development:** The second criterion in choosing a microcontroller is how easy it is to develop products around it. Key considerations include the availability of compiler, emulator, and technical support.
3. **Availability:** The third criterion in choosing a microcontroller is its ready availability in needed quantities both now and in the future. Notice that companies like Microchip Technology and NXP have all dedicated massive resources to ensure wide and timely availability of their products because their products are stable, mature, and single sourced.

Mechatronics and microcontrollers

The microcontroller is playing a major role in an emerging field called *mechatronics*. Here is an excellent summary of what the field of mechatronics is all about, taken from the website of Newcastle University (http://mechatronics2004.newcastle.edu.au/mech2004), which holds a major conference every year on this subject:

"Many technical processes and products in the area of mechanical and electrical engineering show an increasing integration of mechanics with electronics and information processing. This integration is between the components (hardware) and the information-driven functions (software), resulting in integrated systems called mechatronic systems.

The development of mechatronic systems involves finding an optimal balance between the basic mechanical structure, sensor and actuator implementation, automatic digital information processing and overall control, and this synergy results in innovative solutions. The practice of mechatronics requires multidisciplinary expertise across a range of disciplines, such as: mechanical engineering, electronics, information technology, and decision making theories."

Review Questions

1. True or false. Microcontrollers are normally less expensive than microprocessors.
2. When comparing a system board based on a microcontroller and a general-purpose microprocessor, which one is cheaper?
3. A microcontroller normally has which of the following devices on-chip?
   (a) RAM (b) ROM (c) I/O (d) all of the above
4. A general-purpose microprocessor normally needs which of the following devices to be attached to it?
   (a) RAM (b) ROM (c) I/O (d) all of the above
5. An embedded system is also called a dedicated system. Why?
6. What does the term *embedded system* mean?
7. List some of the 8-bit microcontrollers.

SECTION 1.2: OVERVIEW OF THE AVR FAMILY

In this section, we first look at the AVR microcontrollers and their features and then examine the different families of AVR in more detail.

A brief history of the AVR microcontroller

The basic architecture of AVR was designed by two students of Norwegian Institute of Technology (NTH), Alf-Egil Bogen and Vegard Wollan, and then was bought and developed by Atmel in 1996.

You may ask what AVR stands for; AVR can have different meanings for different people! Atmel says that it is nothing more than a product name, but it might stand for Advanced Virtual RISC, or Alf and Vegard RISC (the names of the AVR designers).

There are many kinds of AVR microcontroller with different properties. Except for AVR32, which is a 32-bit microcontroller, AVRs are all 8-bit microprocessors, meaning that the CPU can work on only 8 bits of data at a time. Data larger than 8 bits has to be broken into 8-bit pieces to be processed by the CPU. One of the problems with the AVR microcontrollers is that they are not all 100% compatible in terms of software when going from one family to another family. To run programs written for the ATtiny25 on a ATmega64, we must recompile the program and possibly change some register locations before loading it into the ATmega64. AVRs are generally classified into five broad groups: Mega, XMega Tiny, Special purpose, and Classic. In this book we cover the Mega family because these microcontrollers are widely used. Also, we will focus on ATmega328 since it is powerful, widely available, has DIP packages, and its low cost trainer boards (e.g. Arduino UNO) are available which make it ideal for educational purposes.

For those who have mastered the Mega family, understanding the other families is very easy and straightforward. The following is a brief description of the AVR microcontroller.

AVR features

The AVR is an 8-bit RISC microcontroller with Harvard architecture that comes with some standard features such as on-chip program (code) ROM, data RAM, data EEPROM, timers and I/O ports. See Figure 1-2. Most AVRs have some additional features like ADC, PWM, and different kinds of serial interface such as USART, SPI, I2C (TWI), CAN, USB, and so on. See Figures 1-3 and 1-4. Due to the importance of these peripherals, we have dedicated an entire chapter to many of them. The details of the RAM/ROM memory and I/O features of the Mega are given in the next few chapters.

AVR microcontroller program ROM

In microcontrollers, the ROM is used to store programs and for that reason it is called *program* or *code ROM*. Although the AVR has 8MB (megabytes) of program (code) ROM space, not all family members come with that much ROM installed. The program ROM size can vary from 1KB to 384KB at the time of this writing, depending on the family member. The AVR was one of the first microcontrollers to use on-chip Flash memory for program storage. The Flash memory is ideal for fast development because Flash memory can be erased in seconds compared to the 20 minutes or more needed for the UV-EPROM. A discussion of the various types of ROM is given in Chapter 0, if you need to refresh your memory on these important memory technologies.

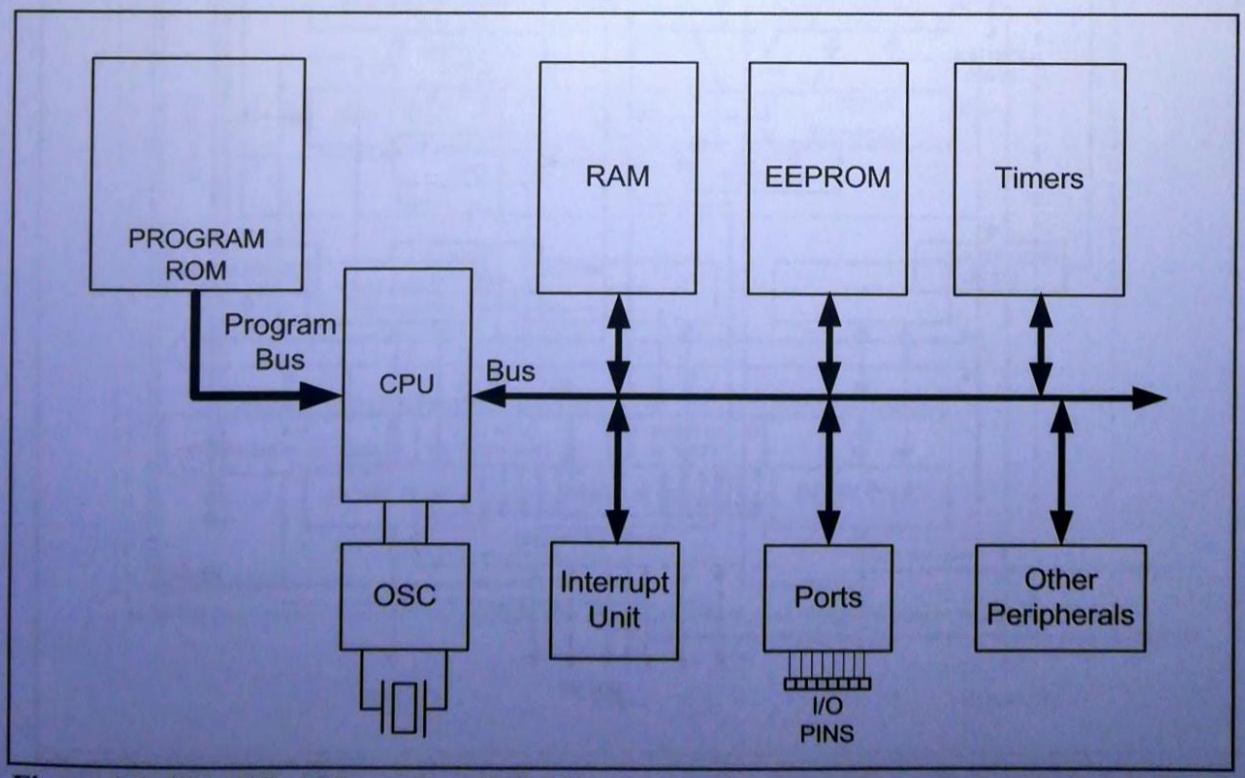

Figure 1-2. Simplified View of an AVR Microcontroller

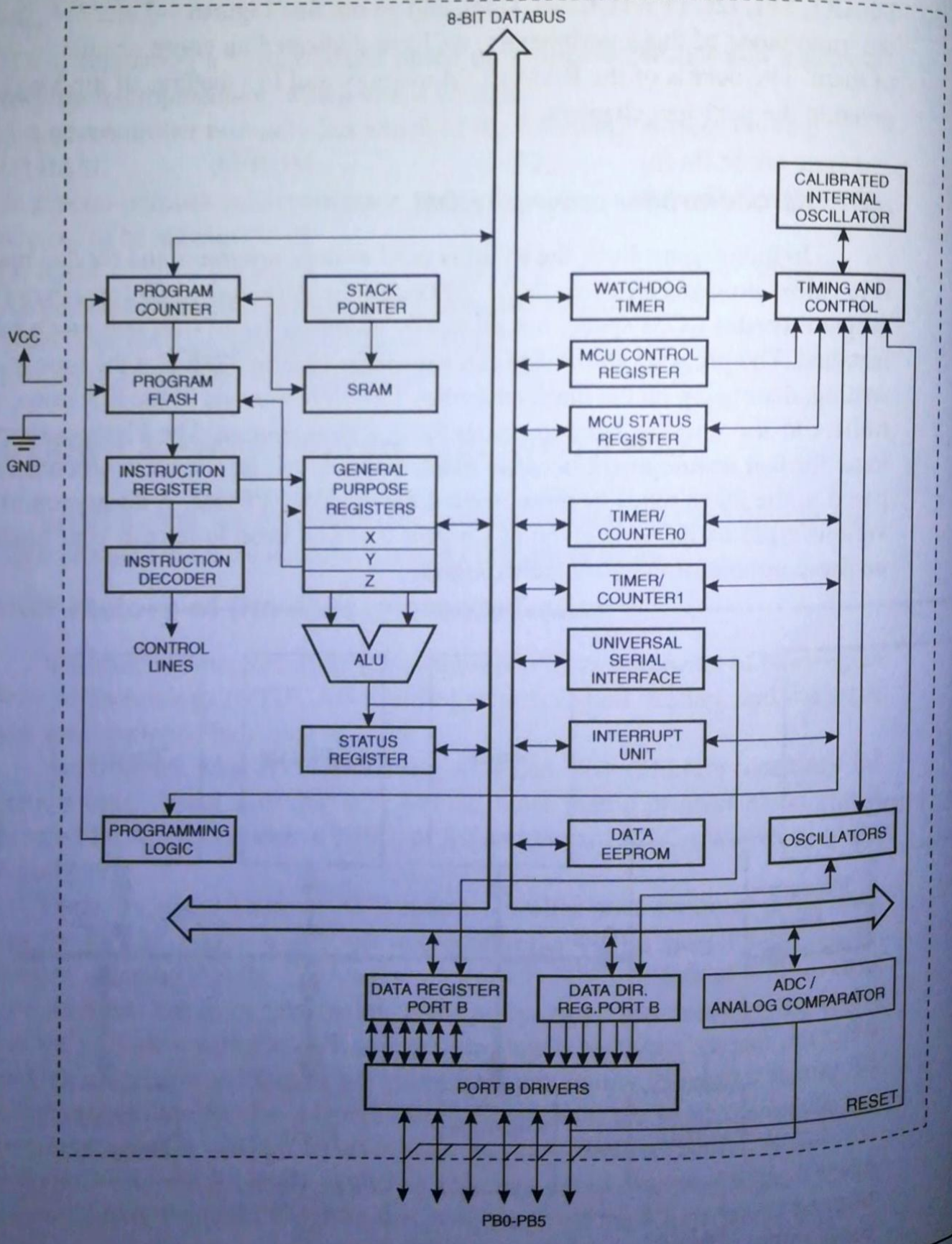

Figure 1-3. ATtiny25 Block Diagram

AVR microcontroller data RAM and EEPROM

While ROM is used to store program (code), the RAM space is for data storage. The AVR has a maximum of 64K bytes of data RAM space. Not all of the family members come with that much RAM. As we will see in the next chapter, the data RAM space has three components: general-purpose registers, I/O memory, and internal SRAM. There are 32 general-purpose registers in all of the AVRs, but the SRAM's size and the I/O memory's size varies from chip to chip. On the

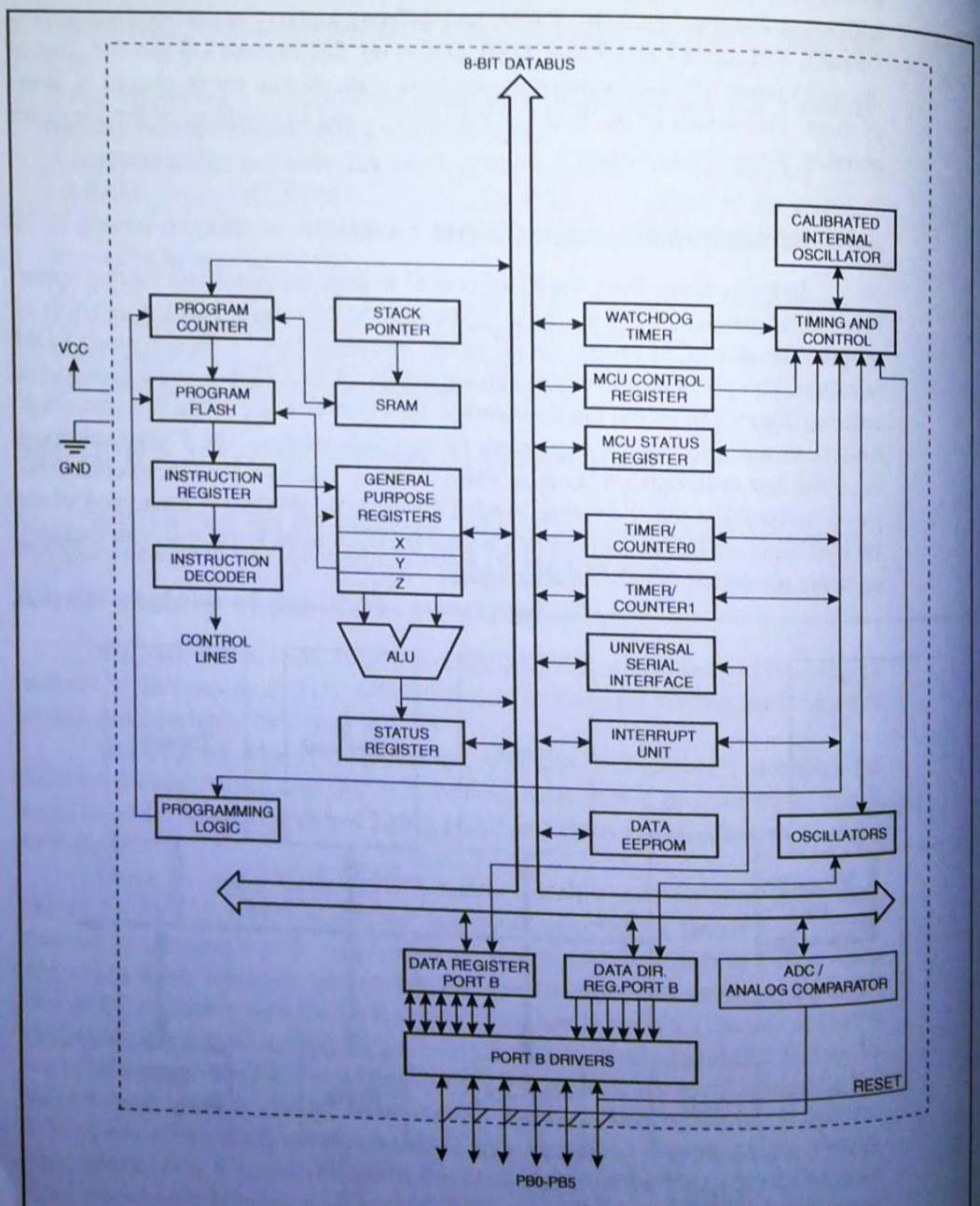

Figure 1-3. ATtiny25 Block Diagram

Atmel website, whenever the size of RAM is mentioned the internal SRAM size is meant. The internal SRAM space is used for a read/write scratch pad, as we will see in Chapter 2. In AVR, we also have a small amount of EEPROM to store critical data that does not need to be changed very often. You will see more about EEPROM in Chapter 6.

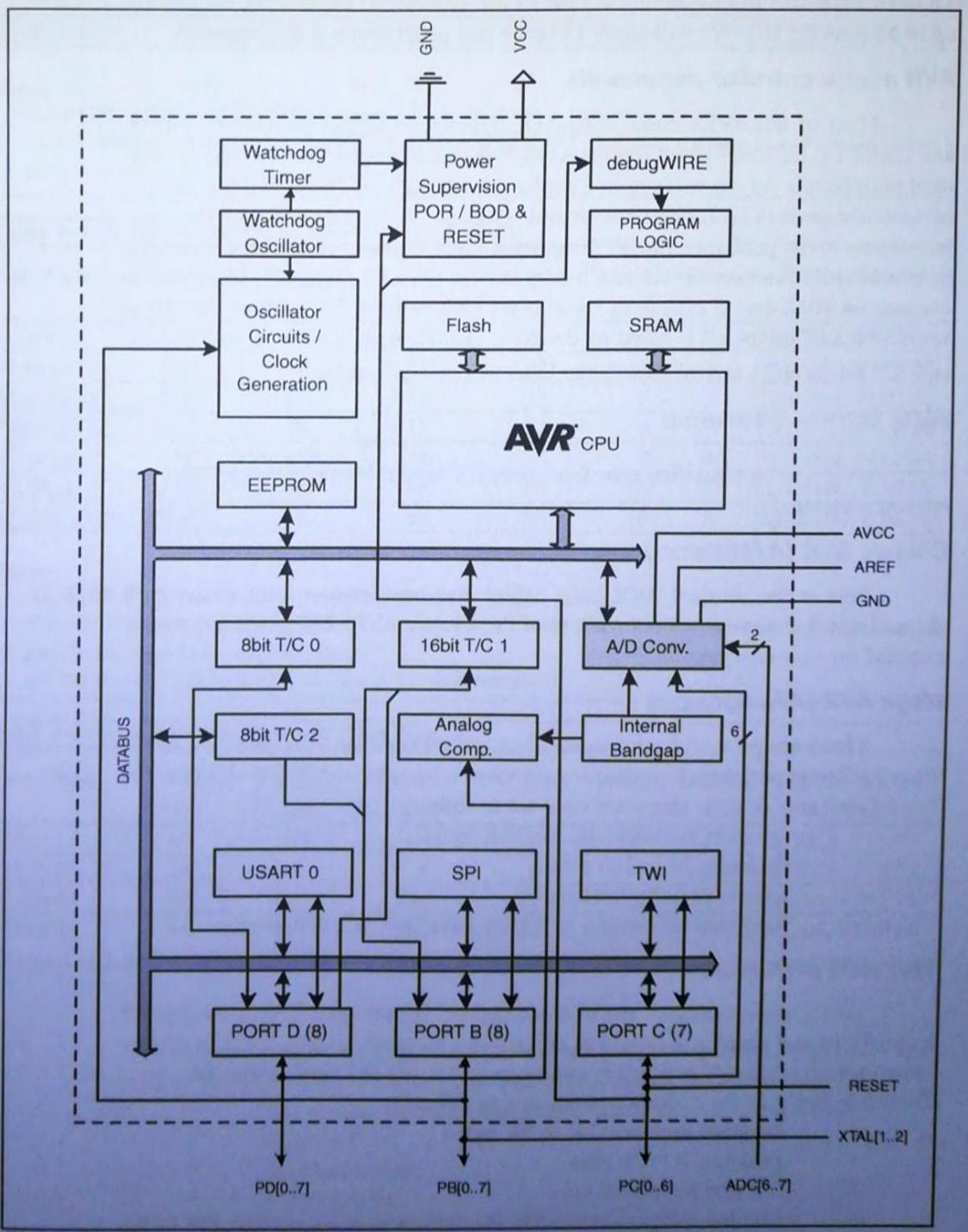

Figure 1-4. ATmega328 Block Diagram

AVR microcontroller I/O pins

The AVR can have from 3 to 86 pins for I/O. The number of I/O pins depends on the number of pins in the package itself. The number of pins for the AVR package goes from 8 to 100 at this time. In the case of the 8-pin AT90S2323, we have 3 pins for I/O, while in the case of the 100-pin ATmega2560, we can use up to 86 pins for I/O. We will study I/O pins and programming in Chapter 4.

AVR microcontroller peripherals

Most of the AVRs come with ADC (analog-to-digital converter), timers, and USART (Universal Synchronous Asynchronous Receiver Transmitter) as standard peripherals. As we will see in Chapter 13, the ADC is 10-bit and the number of ADC channels in AVR chips varies and can be up to 16, depending on the number of pins in the package. The AVR can have up to 6 timers besides the watchdog timer. We will examine timers in Chapter 9. The USART peripheral allows us to connect the AVR-based system to serial ports such as the COM port of the PC, as we will see in Chapter 11. Most of the AVR family members come with the I^2C and SPI buses and some of them have USB or CAN bus as well.

AVR family overview

AVR can be classified into five groups: Classic, Mega, XMega, Tiny, and special purpose.

Classic AVR (AT90Sxxxx)

This is the original AVR chip, which has been replaced by newer AVR chips. Table 1-3 shows some members of the Classic AVR. These are not recommended for new designs.

Mega AVR (ATmegaxxxx)

These are powerful microcontrollers with more than 120 instructions and lots of different peripheral capabilities, which can be used in different designs. See Table 1-4. Some of their characteristics are as follows:

- Program memory: 4K to 256K bytes
- Package: 28 to 100 pins
- Extensive peripheral set
- Extended instruction set: They have rich instruction sets.

Tiny AVR (ATtinyxxxx)

As its name indicates, the microcontrollers in this group have less instructions and smaller packages in comparison to mega family. You can design systems with low costs and power consumptions using the Tiny AVRs. See Table 1-5. Some of their characteristics are as follows:

- Program memory: 1K to 8K bytes
- Package: 8 to 28 pins
- Limited peripheral set
- Limited instruction set: The instruction sets are limited. For example, some of them do not have the multiply instruction.

Table 1-3: Some Members of the Classic Family

Part Num.	Code ROM	Data RAM	Data EEPROM	I/O pins	ADC	Timers	Pin numbers & Package
AT90S2313	2K	128	128	15	0	2	SOIC20, PDIP20
AT90S2323	2K	128	128	3	0	1	SOIC8, PDIP8
AT90S4433	4K	128	256	20	6	2	TQFP32, PDIP28

Notes:
1. All ROM, RAM, and EEPROM memories are in bytes.
2. Data RAM (general-purpose RAM) is the amount of RAM available for data manipulation (scratch pad) in addition to the register space.

Table 1-4: Some Members of the ATmega Family

Part Num.	Code ROM	Data RAM	Data EEPROM	I/O pins	ADC	Timers	Pin numbers & Package
ATmega8	8K	1K	0.5K	23	8	3	TQFP32, PDIP28
ATmega16	16K	1K	0.5K	32	8	3	TQFP44, PDIP40
ATmega32	32K	2K	1K	32	8	3	TQFP44, PDIP40
ATmega328	32K	2K	1K	23	8	3	TQFP32,PDIP28
ATmega64	64K	4K	2K	54	8	4	TQFP64, MLF64
ATmega1280	128K	8K	4K	86	16	6	TQFP100, CBGA
ATmega2560	256K	4K	8K	86	16	6	TQFP100, CBGA

Notes:
1. All ROM, RAM, and EEPROM memories are in bytes.
2. Data RAM (general-purpose RAM) is the amount of RAM available for data manipulation (scratch pad) in addition to the register space.
3. All the above chips have USART for serial data transfer.

Table 1-5: Some Members of the Tiny Family

Part Num.	Code ROM	Data RAM	Data EEPROM	I/O pins	ADC	Timers	Pin numbers & Package
ATtiny13	1K	64	64	6	4	1	SOIC8, PDIP8
ATtiny25	2K	128	128	6	4	2	SOIC8, PDIP8
ATtiny44	4K	256	256	12	8	2	SOIC14, PDIP14
ATtiny84	8K	512	512	12	8	2	SOIC14, PDIP14

XMega AVR (ATmegaxxxx)

These are powerful microcontrollers with some advanced peripheral capabilities. See Table 1-6. Some of their special characteristics are as follows:

- one or two 12-bit ADCs (Other families have 10-bit ADCs.)
- on chip DAC
- DMA controller
- Event system
- AES and DES hardware encryption/decryption which can be used in encrypted communications

Table 1-6: Some Members of the XMega Family

Part Num.	Code ROM	Data RAM	Data EEPROM	I/O pins	USART	Timers	Pin numbe & Packa
ATxmega32A4	32K+4K	4K	1K	34	5	5	TQFP44,QFN
ATxmega256	256K+8K	16K	4K	53	6	7	TQFP64,MLF
ATxmega128A1	128K+8K	8K	2K	78	8	8	TQFP100,BG

Special purpose AVR

The ICs of this group can be considered as a subset of other groups, but their special capabilities are made for designing specific applications. Some of the special capabilities are: USB controller, CAN controller, LCD controller, Zigbee, Ethernet controller, FPGA, and advanced PWM. See Table 1-7.

Table 1-7: Some Members of the Special Purpose Family

Part Num.	Code ROM	Data RAM	Data EEPROM	Max I/O pins	Special Capabilities	Timers	Pin number & Packag
AT90CAN128	128K	4K	4K	53	CAN	4	LQFP
AT90USB1287	128K	8K	4K	48	USB Host	4	TQFP
AT90PWM216	16K	1K	0.5K	19	Advanced PWM	2	SOIC
ATmega169	16K	1K	0.5K	54	LCD	3	TQFP64, MLF

AVR product number scheme

All of the product numbers start with AT, which stands for Atmel. Now, look at the number located at the end of the product number, from left to right, and find the biggest number that is a power of 2. This number most probably shows the amount of the microcontroller's ROM. For example, in ATmega**128**0 the biggest power of 2 that we can find is 128; so it has 128K bytes of ROM. In ATtiny**44**, the amount of memory is 4K, and so on. Although this rule has a few exceptions such as AT90PWM216, which has 16K of ROM instead of 2K, it works in most of the cases.

Review Questions

1. Name three features of the AVR.
2. The AVR is a(n) _____-bit microprocessor.
3. Name the different groups of the AVR chips.
4. Which group of AVR has smaller packages?
5. Give the size of RAM in each of the following:
   (a) ATmega32 (b) ATtiny25
6. Give the size of the on-chip program ROM in each of the following:
   (a) ATtiny84 (b) ATmega328 (c) ATtiny25

SUMMARY

This chapter discussed the role and importance of microcontrollers in everyday life. Microprocessors and microcontrollers were contrasted and compared. We discussed the use of microcontrollers in the embedded market. We also discussed criteria to consider in choosing a microcontroller such as speed, memory, I/O, packaging, and cost per unit. The second section of this chapter described various families of the AVR, such as Mega and Tiny, and their features. In addition, we discussed some of the most common AVR microcontrollers such as the ATmega328 and ATtiny25.

PROBLEMS

SECTION 1.1: MICROCONTROLLERS AND EMBEDDED PROCESSORS

1. True or False. A general-purpose microprocessor has on-chip ROM.
2. True or False. Generally speaking, a microcontroller has on-chip ROM.
3. True or False. A microcontroller has on-chip I/O ports.
4. True or False. A microcontroller has a fixed amount of RAM on the chip.
5. What components are usually put together with the microcontroller onto a single chip?
6. Intel's Pentium chips used in PCs need external _____ and _____ chips to store data and code.
7. List three embedded products attached to a PC.
8. What kinds of computer do exist?
9. True or False. Embedded systems are the largest class of computers.
10. Why would someone want to use a general-purpose microprocessor as an embedded processor?
11. Give the name and the manufacturer of some of the most widely used 8-bit microcontrollers.
12. In a battery-based embedded product, what is the most important factor in choosing a microcontroller?
13. In an embedded controller with on-chip ROM, why does the size of the ROM matter?

SECTION 1.2: OVERVIEW OF THE AVR FAMILY

14. What is the advantage of Flash memory over the other kinds of ROM?
15. The ATmega328 has ____ pins for I/O.
16. The ATmega328 has _____ bytes of on-chip program ROM.
17. The ATtiny44 has _____ bytes of on-chip data RAM.
18. The ATtiny44 has _____ ADCs.
19. The ATmega64 has _____ bytes of on-chip data RAM.
20. The ATmega1280 has ____ on-chip timer(s).
21. The ATmega328 has ____ bytes of on-chip data RAM.
22. The ATmega328 has _____ bytes of data EEPROM.
23. Check the Atmel website to see if there is a RAMless version of the AVR. Give