21609

BOOK 1

Z-80 MICROPROCESSOR PROGRAMMING & INTERFACING

BY JOSEPH C. NICHOLS, ELIZABETH A. NICHOLS AND PETER R. RO

Z-80 Microprocessor Programming & Interfacing

Book 1

by
Elizabeth A. Nichols, Joseph C. Nichols,
and Peter R. Rony

Howard W. Sams & Co., Inc.
4300 WEST 62ND ST. INDIANAPOLIS, INDIANA 46268 USA

FIRST EDITION
THIRD PRINTING—1981

International Standard Book Number: 0-672-21609-4
Library of Congress Catalog Card Number: 79-63822

Printed in the United States of America.

Preface

The microelectronics revolution is here, and gaining momentum. It all began 30 years ago with the development of the transistor. The transistor, a physically small, low-power amplifier, replaced the large, power-hungry vacuum tubes of the first generation computers. Due to a natural synergism between transistors and digital logic, their small size and low cost, transistors have become the basic building blocks for computer circuits. Transistors combine to form gates; gates combine to form flip flops, counters, adders, and other logic functions; and these, in turn, combine to form the memory, control, arithmetic, and logic units which make up the central processing unit (cpu) of a computer. Thus, the number of transistors in a logic circuit has become a reasonable measure of its functional complexity. In 1959, the first integrated circuits consisting of small groups of planar transistors were developed on thin wafers of silicon or germanium. This began the era of Small Scale Integration (SSI) in which 12 or fewer gates could be incorporated into a single integrated circuit (IC). Since 1959, the number of transistors in advanced ICs has been at least doubling every year. Today, circuits containing 262,144 elements are available and the technology is still far from its theoretical limits. The Z-80 CPU and support chips, introduced by Zilog in 1976, represents the state-of-the-art in 8-bit microprocessors. Zilog is currently developing a successor to the Z-80 line, the Z-8000 series of cpu and support chips. However, the Z-8000 will be a 16-bit cpu with computational capacity comparable to mid-range minicomputers, a significant jump in capability. And this is only the beginning. The real revolution will be manifest in the exponential proliferation of products and services dependent on microelectronics.

This book is one of two volumes on Z-80 microprocessor programming and interfacing. Book 1 is on Z-80 software—assembly and machine language programming. Book 2 covers interfacing digital circuits with the Z-80 CPU, PIO, and CTC chips. These books are laboratory oriented texts that are designed to give an integrated approach to microcomputer programming and interfacing. The strong emphasis is on learning through experimentation. Each topic introduced is reinforced with laboratory work that shows not only how ideas succeed, but also where they fail, and what the pitfalls are.

Book 1 requires no background in computer science, programming, or digital electronics. Book 2 however, assumes familiarity with the topics covered in Book 1. In both books, topics are presented in the order that the authors feel is most conducive to learning in a self-study environment. Answers are provided for all the exercises, and every attempt is made to anticipate questions and logical extensions to the experiments.

To enhance the laboratory orientation in the books, the experiments use a sophisticated Z-80–based single-board microcomputer manufactured by SGS-ATES, called the Nanocomputer. The Nanocomputer is an excellent educational computer because it is simple for a novice to use, but incorporates enough options, flexibility, expandability, and sophistication to keep the interest of the most experienced user. For more information on the Nanocomputer, contact SGS-ATES Semiconductor Corp., 240 Bear Hill Road, Waltham, MA 02154.

The authors are indebted to many members of the staff at SGS-ATES in Milano, Italy: R. Baldoni, A. Cattania, B. Facchi, F. Luraschi, C. Wallace, and especially A. Watts whose many ideas and technical expertise on the Nanocomputer tremendously improved these books. Also we wish to thank C. Edson and U. Broggi of SGS-ATES in the USA who greatly expedited progress by acting as liasons between the US and Italian efforts on this project. Finally, much credit is due to J. Titus and D. Larsen of the Blacksburg Group for their efforts in coordinating with Howard W. Sams & Co., Inc. to bring about the publication of these books.

ELIZABETH A. NICHOLS
JOSEPH C. NICHOLS
PETER R. RONY

Contents

CHAPTER 1

Digital Codes

INTRODUCTION

Before you begin to program your microcomputer, it is necessary that you understand how to convert 8-bit binary numbers into hexadecimal code, and vice versa, as well as know certain basic facts about digital codes.

OBJECTIVES

At the completion of this chapter, you will be able to do the following:

- Discuss what is meant by the term *communication.*
- Define *bit.*
- Define *binary code.*
- Define *digital code.*
- Define *hexadecimal code.*
- Convert an 8-bit binary number into a two-digit hexadecimal number.
- Convert a two-digit hexadecimal number into a binary number.
- Distinguish between the binary, hexadecimal, and decimal counting systems.
- List several different digital codes.
- List several different two-state devices.
- Provide one example where the quantity, bits per second, is a measure of information flow.

LANGUAGES, COMMUNICATIONS, AND INFORMATION

One of the most important characteristics that any biological organism (higher order animals) possesses is the ability to communicate with other organisms of the same species. The ability to communicate, which gives many animal organisms a definite survival advantage—in the Darwinian sense of the term—is found in most multicellular creatures, starting with insects and progressing to man. With insects, there exist several modes of communication, including the dance of the bee and forms of chemical communication through remarkable chemical agents called *pheromones*. Man can communicate with the aid of his five senses, as illustrated by handicapped individuals who have lost one or more of their senses but are, nevertheless, highly communicative with those remaining.

Assuming that an individual wishes to communicate with another through the sense of hearing and the use of speech, it is clear that there must be some general agreement concerning how a spoken sound will be interpreted by the individual who hears it. Over the centuries, different regions around the world have each developed their own consensus regarding the meaning of specific sounds and their transcription onto paper. We call such a consensus a *language* or, perhaps, a *foreign language*. Thousands of different languages exist, although only a relatively modest number of them are in widespread use. The popularity of a specific language may wax and wane over the course of several hundred years. Latin, once a dominant language in Europe, is now considered to be a "dead" language, however, it clearly has influenced most of the European languages in very profound ways.

Communication can be defined as the imparting, conveying, or exchanging of ideas, knowledge, information, etc. (whether by speech, writing, or signs).[1]* It is one of the most important and characteristic activities of mankind. As pointed out by James Martin in his excellent book, *Telecommunications and the Computer,*[3] the capacity of major telecommunication links, as measured by a quantity called *bits per second,* has paralleled the advance of civilization over the past one hundred years. The capacity of such links has changed from a rate of 1 bit/second in 1840 to 50,000,000 bits/second in 1970, i.e., a doubling every 5.08 years. Martin has also pointed out that the sum total of human knowledge changed very slowly prior to the relatively recent beginnings of scientific thought. By 1800, it has been estimated that the sum total was doubling every 50 years; by 1950, doubling every 10 years; and that by 1970, it will be doubling every 5 years.

A *language,* which can be defined as the whole body of words and of methods of combination of words used by a nation, people, or race,[1] is just one form of communication. Egyptian hieroglyphics, choreographic scores, mathematical symbols and equations, Ameri-

* See Appendix G for all references.

can Indian smoke signals, the sign language employed by the deaf, and the Morse code are other forms of communication used by man.

BINARY CODING

The "information explosion" would have inundated mankind, at least in the more advanced countries, had it not been for the use of *Two-State Coding* to represent all kinds of information, such as the ten decimal numerals (0 through 9), the twenty-six letters of the English alphabet (A through Z), operations, symbols, motions, and the like. We call such two-state coding *Off-On* or *binary coding.* Binary coding can be represented or manifested by any type of two-state device, such as an on or off light, an open or closed switch, a punched or nonpunched computer card, a "north" or "south" magnetized magnetic core or region of magnetic tape or disc; two different voltage levels, two different current levels, two different frequencies; the words YES and NO; or the abstract symbols 0 (off) and 1 (on). The importance of binary coding resides in the fact that it is possible to construct devices that will change state very quickly, in times as fast as 5 nanoseconds (0.000000005 second). Such a device could, in principle, manipulate, transmit, or receive information at the rate of 200 million bits per second. Thirty-two such devices, operating simultaneously, could manipulate 6.4 billion bits per second. This is the basic capability that has permitted society to store, manipulate, and communicate enormous quantities of information.

BIT

The elementary unit of information is called the *bit,* which is an abbreviation for **BI**nary digi**T**. You can think of a bit as being a light bulb that can be lit (on) or unlit (off) at any given time. Thus, a bit can be pictured as a light bulb that is ON or a light bulb that is OFF. Rather than drawing pictures of light bulbs, we can represent each bulb that is in the lit state by the symbol 1 and each bulb in the unlit state by the symbol 0.

So, a bit is equal to one binary decision, or the designation of one of two possible and equally likely values or states (such as 0 or 1).

Information is typically represented by a series of bits. Thus,

```
1 0 0 0
```

represents decimal 8 in binary code. The series of bits,

```
1 1 0 0 0 0 0 1
```

represents the letter A in 8-bit ASCII code. We shall discuss these two codes shortly.

DIGITAL CODES

A *digital code* is defined as a system of symbols that represent data values and make up a special language that a computer or a digital circuit can understand and use.[3] Digital codes can be considered to be the digital "languages" that permit information to be stored, manipulated, and communicated. Just as there are numerous spoken languages, there also exists a variety of digital codes. Such codes can be subdivided into several important categories:

Category 1. Codes employed by electronic circuitry to perform various digital operations. Example: binary code.

Category 2. Codes employed to convert the decimal numbers 0 through 9 into digital form. Examples: binary code, binary coded decimal (bcd), and gray code.

Category 3. Codes employed to convert decimal numbers, the 26-letter English alphabet, symbols, and operations into digital form. Examples: ASCII code, EBCDIC code, and Baudot code.

Category 4. Instruction codes employed by large computers, minicomputers, and microcomputers that cause the computers to perform a prescribed sequence of operations. Examples: IBM 370 instruction code, PDP 8/E instruction code, Z-80 instruction code.

In this series of modules, we shall pay particular attention to four codes: binary code, binary coded decimal (bcd), ASCII code, and the instruction code for the Z-80 microprocessor chip.

BINARY CODE

The simplest digital code is a two-state, or binary, code that consists of a 0 (off) and a 1 (on) state. We call these two states *logic 0* and *logic 1*. In binary code, decimal 0 is represented by a logic 0 and decimal 1 by a logic 1. This should be quite clear. How, on the other hand, are higher decimal numbers, such as 3, 17, 568, etc., represented using binary code? The answer is that we use a series of bits to build a *binary counting system* that is formed on a *base*, or *radix*, of two. For example, the binary number 11101_2, where the subscript (2) represents the binary counting system, is equivalent to

```
11101 (2) = (1×2**4) + (1×2**3) + (1×2**2) + (0×2**1) + (1×2**0) = 29 (10)
```

where you should keep in mind that A**B is equivalent to A^B. Therefore,

$$\begin{aligned} 2{*}{*}4 &= 16 \text{ in decimal notation} = 16_{10} \\ 2{*}{*}3 &= 8 \text{ in decimal notation} = 8_{10} \\ 2{*}{*}2 &= 4 \text{ in decimal notation} = 4_{10} \end{aligned}$$

$$2**1 = 2 \text{ in decimal notation} = 2_{10}$$
$$2**0 = 1 \text{ in decimal notation} = 1_{10}$$

Therefore,

11101 (2) = 16 (10) + 8 (10) + 4 (10) + 0 + 1 (10) = 29 (10)

where the subscript (10) associated with these numbers represents the decimal counting system, a system that is formed on a base, or radix, of 10. A brief table follows that allows you to convert simple decimal numbers into binary numbers.

Decimal Number	Binary Number
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000

Thus, a series of four binary digits, or bits, can represent any of sixteen different decimal numbers ranging from zero to fifteen. Decimal numbers larger than fifteen require additional bits, as shown in the following table:

Decimal Number	Binary Number
0	0
1	1
2	10
3	11
4	100
7	111
8	1000
15	1111
16	10000
31	11111
32	100000
63	111111
64	1000000
127	1111111
128	10000000
255	11111111
256	100000000

511	111111111
512	1000000000
1023	1111111111
1024	10000000000
2047	11111111111
2048	100000000000
4095	111111111111
4096	1000000000000
8191	1111111111111
8192	10000000000000
16,383	11111111111111
16,384	100000000000000
32,767	111111111111111
32,768	1000000000000000
65,535	1111111111111111

Therefore, an 8-bit binary number can encode two hundred and fifty-six different decimal numbers, ranging from 0 to 255_{10}, or two hundred and fifty-six different "things," no matter what they may be (instructions, devices, pulses, etc.). The Z-80 is a microprocessor chip that has a 16-bit memory address and an 8-bit I/O device word. This means that it can directly address 65,536 different memory locations and can generate at least 256 different I/O pulses or device addresses.

HEXADECIMAL (HEX) CODE

It can be difficult to remember binary numbers that contain many bits. For example, can you remember the following 8-bit binary number,

```
1 0 0 1 1 1 0 1
```

after having looked at it for only one second? Quick, cover it up or look away! Consider also the problem of remembering a list of such 8-bit numbers:

```
1 1 0 1 1 0 1 0
1 1 1 0 0 1 0 1
0 1 1 0 1 0 0 1
1 0 1 0 1 0 1 1
```

You probably will conclude that there must be a better way to remember 8-bit binary numbers. We are using 8-bit numbers here because you will encounter them frequently when you begin to program the 8-bit Z-80 microcomputer.

One approach to remembering multi-bit binary numbers is the use of *hexadecimal code*. The term *hex* is simply an abbreviation for the word *hexadecimal*. Hexadecimal code refers to the *hexadecimal counting system,* a system that is formed on a base, or radix, of 16. The hexadecimal counting system consists of sixteen different symbols: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E, and F. Just as we did with decimal numbers, it is possible to convert hexadecimal numbers into binary numbers:

Decimal Number	Hex Number	Binary Number
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	0001 0000
17	11	0001 0001
18	12	0001 0010
19	13	0001 0011
20	14	0001 0100
21	15	0001 0101
22	16	0001 0110
23	17	0001 0111
24	18	0001 1000
32	20	0010 0000
40	28	0010 1000
48	30	0011 0000
56	38	0011 1000
63	3F	0011 1111

We have grouped the 8-bit binary numbers into two groups of four bits each to help you understand how the hexadecimal number to binary number conversion was made. While the space between each 4-bit group does not affect the value of the number, it does make the binary number easier to read and has become a standard convention.

We now address the question of how to convert an 8-bit binary number into hex code. The procedure to accomplish this conversion requires three steps:

1. Write down the full 8-bit binary number.
2. Split this 8-bit binary number into two groups with four binary digits in each group.
3. Substitute the equivalent hex digit

   0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

   for each group of four bits.

Having done this you will have converted an 8-bit binary number into a two-digit hex code. Each group of four binary digits is converted independently of the other.

As an example, consider the 8-bit binary number,

1 0 0 1 1 1 0 1

First, split this binary number into two groups of four binary digits each

1001 1101

Finally, substitute the equivalent hex digit for each of these two groups.

9 D

This is the correct answer, 9D (16), where the subscript (16) means "relative to" the hexadecimal counting system. Some additional hex numbers and their corresponding 8-bit binary numbers are listed below:

Decimal Number	Binary Number	Hex Number
64	0100 0000	40
72	0100 1000	48
73	0100 1001	49
74	0100 1010	4A
96	0110 0000	60
120	0111 1000	78
127	0111 1111	7F
128	1000 0000	80
160	1010 0000	A0
184	1011 1000	B8
191	1011 1111	BF
248	1111 1000	F8
255	1111 1111	FF

A NOTE ON NOTATION

It may have occurred to you that dealing with all of these different methods of number representation—binary, hex, and decimal—that there is a possibility for some confusion. For example, the number 10 can be a decimal or a hex or a binary number. To remedy this problem, whenever there is any possibility for ambiguity, all hexadecimal numbers will be followed by the letter H, e.g., 10H, all decimal numbers will be followed by a period or decimal point, e.g., 10., and all binary numbers will appear without any special notation, e.g., 10 or 0110.