

PROGRAMMING MICROCOMPUTERS

WITH SAMPLE PROGRAMS

STANLEY J. EVANS

Preface

The development of microcomputers has given birth to a new era in the electronic world. The impact of the microprocessor is threefold:

1. Cheap, digital computing capability can be placed anywhere the designer wants it.
2. For any digital system, microprocessors can greatly reduce component count.
3. Engineering turnaround time is cut drastically.

These simple truths account for the current explosion in microprocessor activity. And explosion is just what it is. Almost all new data-processing systems at or below what one could call the minicomputer level of complexity are being designed with microprocessors. Furthermore, the availability of cheap computing power enables designers to incorporate decision-making capability into applications never considered before.

However, as true as it is that microprocessors reduce hardware development time, it is also true that the tough task for the designer is now in programming and debugging. Even for the most experienced, software development is tricky. Learning to pick the right language level and how to work with it efficiently is essential.

To varying degrees, engineers must become programmers when working with processors. For the more complex systems, it is likely that they will share the work with professional programmers—computer specialists drawn into the field by the rapid boom in programming. Even then, they will need to learn the basics in order to take care of the peripheral-control tasks off-loaded from the central processing unit.

The text has been written for readers with little or no experience in programming. However, it has taken for granted that the reader is familiar with many binary functions and digital operations. Part I covers theory, with examples and exercises; while Part II provides program examples and other useful reference material.

I wish to extend my gratitude to the various manufacturers such as National Semiconductors, Texas Instruments, Intel, and the others who have helped me with their guidance to write this text.

Stanley J. Evans

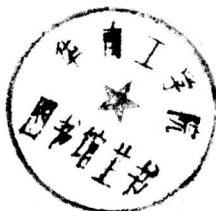
TP36 TP31
E/3 E2

8064909

Contents



E8064909



PREFACE, ix

Part One

MICROPROGRAMMING THEORY AND EXAMPLES

Chapter One

Basic Concepts, 3

- 1.1 Introduction, 3
- 1.2 Description of Hardware, 3
- 1.3 Use of a System, 5
- 1.4 Program Development, 6

Chapter Two

Flowcharting, 7

- 2.1 General, 7
- 2.2 Symbols Used in Flowcharting, 8
- 2.3 Design of Flowcharts, 8

Chapter Three

Introduction to Microprocessors and Their Terminology, 14

- 3.1 Introduction, 14
- 3.2 Basic System, 14

	3.3 System Operation, 16
	3.4 Programmable Clocks, 18
	3.5 Direct Memory Access, 18
Chapter Four	Computer Arithmetic, 21
	4.1 Introduction, 21
	4.2 The Decimal System, 22
	4.3 The Octal System, 22
	4.4 The Binary System, 25
	4.5 Binary Coded Decimal (BCD), 29
	4.6 The Hexadecimal System, 30
	4.7 Program Coding, 32
	4.8 ASCII Code, 35
Chapter Five	Programming Techniques and Terminology, 37
	5.1 Introduction, 37
	5.2 Programming Steps, 38
	5.3 Memory Requirements, 39
	5.4 Terminology, 40
	5.5 Software Design Considerations, 46
	5.6 Addressing Modes, 47
	5.7 Instruction Types and Instruction Sets, 48
Chapter Six	Programming Languages, 55
	6.1 Introduction, 55
	6.2 Machine Language and Assembly Language, 55
	6.3 FORTRAN, 58
	6.4 COBOL, 60
Chapter Seven	BASIC
	7.1 Introduction, 61
	7.2 Basic Terminology, 61
	7.3 Loops, 71
	7.4 Subroutines, 72
	7.5 The Input Statement, 76
	7.6 Random Numbers, 77
Chapter Eight	Advanced BASIC
	8.1 Introduction, 80
	8.2 Strings, 80

- 8.3 Arrays, 82
- 8.4 Vectors and Matrices, 85

Chapter Nine

PL/M, 95

- 9.1 Introduction, 95
- 9.2 The Organization of a PL/M Program, 95
- 9.3 Basic Constituents of a PL/M Program, 97
- 9.4 Identifiers and Reserved Words, 98
- 9.5 Comments, 99
- 9.6 PL/M Statement Organization, 99
- 9.7 PL/M Data Elements, 100
- 9.8 Well-Formed Expressions and Assignments, 102
- 9.9 DO Groups, 106
- 9.10 Subscripted Variables and the Initial Attribute, 109
- 9.11 A Sorting Program, 111
- 9.12 Procedure Definitions and Procedure Calls, 112
- 9.13 Based Variables, 116
- 9.14 Long Constants, 119
- 9.15 Scope of Variables, 122
- 9.16 Statement Labels and GO TO's, 124
- 9.17 Compile-Time Macro Processing, 129
- 9.18 Predeclared Variables and Procedures, 130
- 9.19 Compiling and Debugging PL/M Programs, 133
- 9.20 PLM1 Operating Procedures, 134
- 9.21 PLM2 Operating Procedures, 143
- 9.22 Program Checkout, 145
- 9.23 Implementation-Dependent Operating Procedures, 158
- 9.24 PL/M Run-Time Conventions for the 8008 CPU, 166
- 9.25 Subroutine Linkage Conventions, 169
- 9.26 Use of Assembler Language Subroutines with PL/M, 170

Chapter Ten

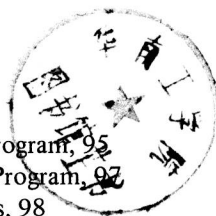
APL, 173

- 10.1 Introduction, 173
- 10.2 What is APL? 173
- 10.3 Interpretation of Operators, 178

Part Two

USEFUL PROGRAMS IN BASIC AND REFERENCE TABLES

- ASCII Codes, 187
- Hexadecimal Arithmetic, 188



	Powers of Two, 189
	Powers of Sixteen, 190
	Powers of Ten, 190
	Hexadecimal-Decimal Integer Conversion, 191
	ASCII Conversion Code, 200
	128 Character ASCII Table, 201
	Number Base Conversion Program in BASIC, 202
	Dow-Jones Industrial Average Forecaster, 203

Appendix I	List of Microprocessor Manufacturers, 205
------------	--

Appendix II	Glossary of Microprocessor Terminology, 207
	Index, 225

8064909

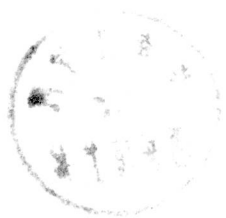
PART ONE

**MICROPROGRAMMING
THEORY AND EXAMPLES**



1911

1911



CHAPTER ONE

Basic Concepts

1.1 INTRODUCTION

Programming is the means for communication between a computer and its human user: microprogramming is designed specifically for use with microcomputers.

The chapters that follow provide thorough discussion on the terminology and the techniques used in microprogramming. However, one cannot rely on texts alone to obtain a thorough knowledge of the subject: the only way to really learn microprogramming is to actually practice it with a microcomputer. A low-priced microcomputer system to be used either as a hobby or in one's profession would be well worth the investment. A basic microcomputer system is shown in Fig. 1-1, and all sample microprograms and exercises in the text have been designed around and tested with such a system. A list of suppliers and descriptions of the systems they offer will be found in Appendix I.

1.2 DESCRIPTION OF HARDWARE

The basic system shown in Fig. 1-1 consists of three units, as follows:

1. CRT (display). This unit is the visual means for communication between the computer and the user. When a program is fed into

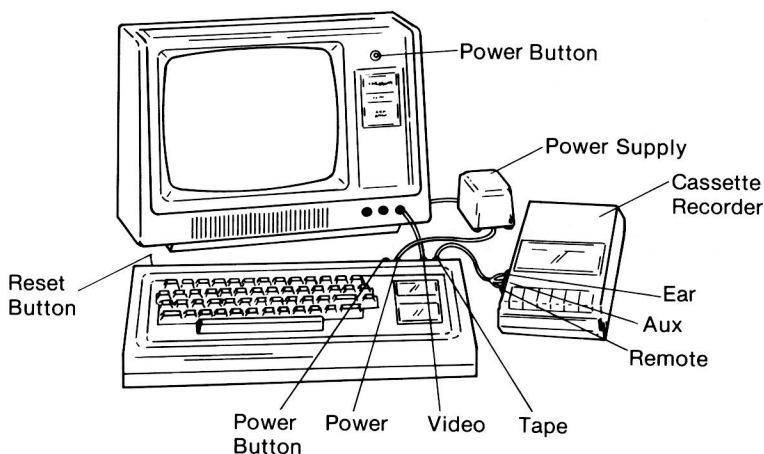


FIG. 1-1 Basic microcomputer system.

the system through the keyboard, it will be displayed on the screen and the results of the program will also be displayed on the same screen. Other computer systems use CRT displays and different peripheral equipment for display purposes.

2. **Keyboard-CPU.** This unit is the heart and brain of the basic computer system. Any program fed into the system is typed on the keyboard, goes through the central processing unit and other memory components, and, at the same time, is displayed on the CRT. The keyboard unit performs all the work required to analyze the program and arrive at a result. Any additional peripheral units (such as a tape recorder or a printer) are also controlled by this unit.

The keyboard of the unit, with the exception of several specially marked keys, resembles that of an ordinary typewriter. The keyboard is frequently mentioned throughout the text, and a description of each key function is provided wherever a particular key is used. The pictorial diagram of a keyboard unit is shown in Fig. 1-2. At this stage, you need not be concerned with the electronic function of the keyboard but need only to remember how a key functions in a particular program.

3. **Tape recorder.** Although this unit is very useful in a system, it is entirely optional: the system can operate very well without a

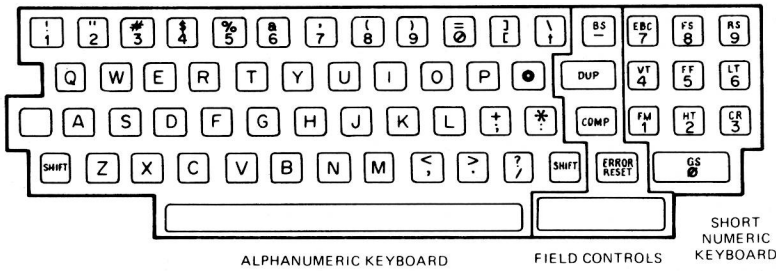


FIG. 1-2 Microcomputer keyboard. (Courtesy, Olivetti)

recorder. The unit is used to feed prerecorded programs into, and record programs from, the system. One advantage that a recorder offers is speed. It usually takes 3 minutes for a basic prerecorded program to be fed into the system: this is about 1/3 the amount of time that it would take to type the program into the system. Another advantage is that the user may record programs and use them in future applications.

The basic system may be expanded to include as many peripherals as the user may desire or can afford. However, the system described above is more than adequate for training in the art of microprogramming.

1.3 USE OF A SYSTEM

A computer system operates as well, or as badly, as its user programs it to operate. Computers are a wonderful invention, but remember, they have been designed by humans and still depend on human ingenuity for proper performance. When a program is poorly designed, the results will be accordingly poor.

A basic system that offers expansion capabilities would be the most practical system for someone who intends to start with basic operations and gradually get involved in more sophisticated programming.

1.4 PROGRAM DEVELOPMENT

A well-designed program must follow an orderly sequence. First, you must identify the problem you wish to solve. In the problem, identify which factors are known and which are unknown. For example, we are asked to write a program that will determine the time required to travel by train from New York City to Washington, D.C. if the distance is 300 miles and the average speed of the train is 80 miles per hour. In this problem T (time) is unknown, and D (distance) and S (speed) are known.

The second important factor in program development is flowcharting. Had you been asked to write a program for a more complicated problem, you would have had to outline a sequence of events from the beginning of the problem to the arrival at the solution. Flowcharting is more or less a symbolic itinerary of the program and indicates the possible routes of the program.

A third important factor in program development is the ability to communicate with the computer, and this is where microprogramming comes in handy. The user has to know what programming language a computer uses and how to apply that language to a specific program. Various languages, such as FORTRAN, ALGOL, PL/I, and BASIC have been designed for communication with a computer. These are described in Chapters 6 through 10, with emphasis on BASIC because it is the most popular and simplest language.

CHAPTER TWO

Flowcharting

2.1 GENERAL


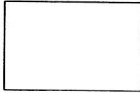
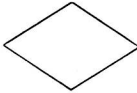
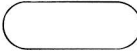

Flowcharting is a general pictorial diagram of the sequence of events a program has to follow. The success of a program depends greatly on how well its flowchart has been designed.

There are usually three different types of flowcharts, as follows:

1. The system flowchart, which describes only the machines involved in the problem solution.
2. The general flowchart, which defines exactly what the problem is and the general plan for solution.
3. The detail flowchart, which illustrates every detail of the method described by the general flowchart.

2.2 SYMBOLS USED IN FLOWCHARTING

The most essential clarification symbols used in flowcharting are described below.

Symbol	Name	Typical Use
	Input/Output	reading files; writing records
	Process	adding, subtracting, multiplying, dividing, moving data
	Decision	comparing two quantities; checking for end of file
	Terminal	beginning the run; stopping the run
	Connector	continuing the program

2.3 DESIGN OF FLOWCHARTS

In order to design flowcharts, you should first acquaint yourself with the usage of the above symbols. They are really fairly simple to memorize.

Each flowchart starts and ends with the terminal symbol. This symbol appears only twice in a flowchart. The words “start” and “end” are written inside the symbol and are not commands to the computer. They simply designate the actual starting and finishing points of the flowchart.

The input/output symbol appears at any point at which data are either to be entered into the system or to be retrieved from it.

The process symbol is used to determine the steps involved in manipulating the data to achieve the desired result.

BASIC, PL/M, and other microcomputer languages are fairly simple, and their flowcharting should not present any problems. However, in larger systems with more complicated programs, flowcharting may be a very strenuous operation.

Let us illustrate a flowchart example. The given problem is to grade a ten-question test by comparing each of the student's answers with the correct answer. We will put the correct answers in a DATA statement (see Appendix II for microprogramming terminology) in the program, enter a student's answer through a keyboard such as the one illustrated in Fig. 1-1, compare (grade) them, then print the percentage of correct answers. This procedure will be repeated until all of the students' papers are graded.

The problem, in terms of flowcharting, may be divided as follows:

1. START-END

Self-explanatory.

2. PROCESS

Here we have several processing steps to follow:

- a. Get ready to grade paper.
- b. Loop 10 times because there are 10 answers. (Looping techniques are described later on.)
- c. Enter student's answer.
- d. Find correct answer.
- e. Add 1 to number correct, and print "CORRECT."

3. DECISION

The following three decisions are appropriate:

- a. Is student's answer correct?
- b. Have we looped 10 times?
- c. Are there any more students?

Our flowchart should contain 1 START, 1 END, 5 PROCESS, and 3 DECISION DIAMONDS. Fig. 2-1 illustrates such a flowchart. Notice the line numbers in each symbol. In BASIC and other microprogramming languages each statement must start with an arbitrary number. The computer uses this number to identify a particular statement.

Although it is too early to get involved in a program, it is worth observing the program that would develop from the flowchart:

```

10  REM *TEST GRADER*
20  CLS
40 P. "ENTER THE STUDENT'S 10 ANSWERS AS REQUESTED"
50  RESTORE

```

```

60  N=0
70  FOR I=1 to 10
80  PRINT "ANSWER NUMBER"; I;
90  INPUT A
100 READ B
110 PRINT A, B;
120 IF A=B THEN PRINT "CORRECT"; N=N+1
130 PRINT
140 NEXT I
150 P.:P.N; "RIGHT OUT OF 10=";
160 PRINT N/10 *100; "%"
170 P.:P. "ANY MORE TESTS TO GRADE";
180 IN. "---1=YES, 2=NO"; Z
190 IF Z=1 GO TO 50
200 DATA 65, 23, 17, 56, 39.

```

Since you are not yet familiar with microprogramming, the above program undoubtedly appears rather peculiar. You are probably uncertain of the meaning of terms such as REM, CLS, P., RESTORE. You can obtain some idea of what the above program calls for, but the terminology and the entire format of the program are extremely unfamiliar. However, not to worry. In the chapters that follow, the entire secret of microprogramming will be laid out before you, and programs such as the above will become mere "child's play" to you.

A simple flowchart such as the above would not suffice for a more complicated problem. For such a problem, we may wish to subdivide a flowchart into larger modules and use a *master flowchart* to indicate the relationship between the flowcharts of the individual programs.

As an example, suppose that we wish to construct a program that calculates the return on various investments. The various investments (or options) might be:

1. Government bonds
2. Savings account
3. Investment in a private company
4. Secured loan to someone
5. Stock market

We would now flowchart each of the individual programs separately. The investment in a private company would, for example, have to contain the rate of return, size of investment, and the number of