JACK JAY PURDUM

BASIC-80 AND CP/M

BASIC-80 and CP/M

Jack J. Purdum

Butler University
Indianapolis, Indiana

Macmillan Publishing Co., Inc.
NEW YORK

Collier Macmillan Publishers
LONDON

Macmillan Publishing Co., Inc.
866 Third Avenue, New York, New York 10022

Collier Macmillan Canada, Inc.

Library of Congress Cataloging in Publication Data

Purdum, Jack J.
BASIC-80 and CP/M.

Includes index

1. Basic-80 (Computer program language)
2. CP/M (Computer program) I. Title.

QA76.73.B33P87 1983 001.64'24 82-16195
ISBN 0-02-397020-0

Printing: 2 3 4 5 6 7 8 Year:3 4 5 6 7 8 9 0

ISBN 0-02-397020-0

BASIC-80© and CP/M®

To my parents,
John and Janette Purdum

PREFACE

The central objective of this book is to teach the reader to program a microcomputer in Basic. But there are dozens of Basic programming books with that same objective. There are two important features about this text, however, that distinguish it from the others.

(1) The book is written for a specific dialect of Basic. This allows specific discussion of the more critical aspects of programming—namely, disk file operations. Many texts ignore disk file operations totally. This text covers conventional sequential and random file structures, plus a hybrid called Skip Sequential.

(2) The book is written for a specific operating system. This aspect of software is usually ignored in other texts, yet no Basic can function without one. An understanding of Basic and the operating system under which it runs is important to the programmer.

The dialect of Basic used in the text is Microsoft's BASIC-80 and the operating system is CP/M by Digital Research. Microsoft has produced the Basic used on most of the popular microcomputers (including the Apple, TRS-80, and the IBM Personal Computer). Digital Research's CP/M is the most widely used operating system in the industry. If the reader finds that his or her microcomputer does not use this combination directly, chances are very good that the computer uses a derivative of the two.

The book is based on the philosophy that the only way to learn programming is to write programs. Each chapter contains several "Things to Do" sections with program problems. The programs in these sections serve as building blocks for learning new concepts. In a formal programming course, these can be used as programming assignments. If the book is used as a self-study guide, the reader is encouraged to work these problems before continuing. Both insight and experience will be gained by working these problems.

A number of chapters are followed by appendixes. The appendix to Chapter 2, for example, contains a discussion of the BASIC-80 program editor. Understanding this editor makes writing and changing subse-

quent programs much easier. Other appendixes provide additional details (and insights) about topics discussed in the chapter.

The book attempts to explain the "how and why" of programming. When programming choices confront the reader, the book explains how and why the options are different. Chapter 4 and its appendix are an example of this. It is hoped that this effort will reinforce the idea that there is a difference between memorizing and truely understanding a concept.

In addition to the normal complement of programming topics, the book also contains chapters on sorting and searching, simulation, and the CP/M operating system. The book concludes with a rather large program that combines most of the Basic commands presented in the text. The reader should find it useful and instructive.

There are a number of people who must be acknowledged: Frank Parker, who taught me what a PRINT statement is; Dr. Marshall Dixon and John Freeland, for my appreciation of the electronics that makes a microcomputer tick; and my friends and colleagues Tim Leslie and Dr. J. B. Orris, for their insights, suggestions, and support. The book reflects the critical eye of the publisher, editor, and a number of anonymous reviewers, and I appreciate their comments and suggestions. Lastly, a special recognition to my wife Karol, who put up with me while the book was being written.

J. J. P.

CONTENTS

1 Where to Begin

Learning Basic is not much different from learning anything else; it takes practice. And practice means sitting down at the computer and trying to write a program. We'll proceed on the assumption that you are sitting in front of the computer and you see the following on the screen:

```
A>
```

This is the prompt that appears on the screen when CP/M®* is in control of the computer. We need not concern ourselves at this point about the details of CP/M; we'll come to that in due time. It does, however, let us "communicate" with the computer. To illustrate this fact, try typing in your name.

If I typed in my name, the following would appear on the screen:

```
A>JACK■
```

The underlined printing is used to represent things that you enter from the keyboard. The block at the end of my name is called a **cursor** and is used to show where the next character is to be placed on the screen. The cursor may be an underline character or it may be a box, as shown here. Whatever it is, the cursor will now be at the end of your name waiting for you to press another key.

Since we've already typed in the name and nothing has happened, the computer must be waiting for us to do something else. If you scan the keyboard, there should be either a RETURN or ENTER key somewhere (usually toward the upper right side of the keyboard); it might be abbreviated as RET or ENT. Now press the RETURN (or its equivalent) key on your keyboard. Unless you have a very unusual name, you will see

*CP/M is a trademark of Digital Research.

```
A>JACK
JACK?
```

on the screen. What we did was issue a command to CP/M named JACK when we pressed the RETURN key. The first thing CP/M did (and did very quickly) was to see if JACK was a valid CP/M command, which it isn't. Since that check failed, CP/M then checks to see if there is a program named JACK on the disk that it could run. Since no such program existed on the disk, CP/M "gave up" trying to execute the command JACK and issued us an error message (i.e., the JACK? that is on the screen).

Since that attempt to communicate with the computer didn't work too well, let's try something else. Now type in

```
A>DIR
```

and press the RETURN key. There should now be some new information on the screen, which might look like the following:

```
A>DIR
A: MBASIC   COM : TEST     BAS : CONSOL   ASM : TIME     BAS
A: SALES    DAT : PARMD    BAK : STAT     COM : PROG1    BAS
A>
```

What we did with the DIR command was ask CP/M to give us a DIRectory of all of the files found on the disk drive named "A." (The "A" in the CP/M prompt "A>" means that we are presently using the disk drive named drive A.)

CP/M responded to the DIR command by displaying an "A:" (which tells us that it is looking on disk drive A), followed by a list of the disk files currently stored on drive A (e.g., MBASIC.COM, TEST.BAS, etc.). Each of these files represents a collection of data that have been previously stored on the disk and may be accessed by the name displayed. We shall discuss some of the details of disk files shortly.

The fact that your name produced an error message and that DIR listed the files on the disk illustrates an important point: There are a limited number of commands that CP/M can understand. If CP/M doesn't understand the command, it will print an error message (usually the command you just entered, followed by a question mark).

If your computer has two disk drives, try entering:

```
A>DIR B:
```

and press the RETURN key. If there was a disk in drive B and the drive was turned on, you will now see the files contained on the disk in drive B listed on the screen. The DIR B: command to CP/M simply requests a DIRectory of all files contained on drive B.

CP/M "understands" several other commands besides DIR; we'll discuss those in later chapters. Before proceeding, however, we need to understand the way CP/M expects to find disk files on the disk. The general form for a CP/M disk file is

filename.filetype

The filename is limited to eight characters or less and is the primary name of the disk file. The "." is used to separate the filename from its filetype. The filetype for a disk file is optional, but usually denotes the generic type of file and is limited to three characters or less. For example, a filetype of BAS is usually used for a BASic program disk file. If we have a Basic game program stored on the disk and its name is DICE, it would appear in the directory listing as DICE.BAS.

Another common type of disk file is the COM-type file. A COM disk file is one that CP/M can load into memory from the disk and execute directly. To list all the COM files that you have stored on your computer, try entering the following command:

```
A>DIR *.COM
```

Press the RETURN key and look closely at the information on the screen. This CP/M command is asking for a DIRectory of all data files that are COMmand-type disk files. The asterisk (*) was used in this command instead of a specific filename. In other words, *the asterisk is used as a universal replacement (or "wild card") for all filenames and allows us to get a directory of all COM files stored on the disk regardless of their specific file name.*

What would happen if we typed in

```
A>DIR *.*
```

and pressed the RETURN key? Since we have used the asterisk for both the filename and filetype, this command to CP/M is requesting a directory of all disk files regardless of specific filename or filetype; it produces the same results as a simple DIR command to CP/M.

THINGS TO DO

1 What will the following CP/M commands do?

```
DIR PROGRAM.*
DIR *.BAS
DIR *.COM
DIR B:*.*
DIR B:*.COM
DIR B:*.BAS
```

Now that you've had a little practice listing the disk files, list all of the COM files on drive A and look for one named MBASIC.COM. This particular COM file is the standard file name for Microsoft's version of Basic, called BASIC-80©.* We shall be using this version of Basic in all of the programming examples in this book. Since it is a COM-type file, it is a disk file that CP/M can load directly into the computer's memory and then execute.

Loading Basic

How do we get Basic from the disk into the computer so that we can use it? Simple—we let CP/M do it for us. The COM at the end of the file name tells CP/M that the file named MBASIC is a COMmand file that can be executed by the computer. All we have to do is give CP/M the file name and let it take care of the details. Try entering:

```
A>MBASIC
```

and press the RETURN key. After a few seconds you should see Microsoft's sign-on message, ending with the word "Ok." When you see "Ok" on the screen, this means that CP/M has given control to BASIC-80 and we are now in the "direct mode" of BASIC-80.

In direct mode, BASIC-80 is in control and we are communicating to the computer through it rather than directly through CP/M. To prove this point, if you try typing in the CP/M DIR command, you will see the following on the screen:

```
DIR
Syntax error
```

which means that BASIC-80 (called B-80 from now on) didn't understand the DIR command that you just entered. Why not? The reason is because DIR is not one of the commands that is understood by B-80. Buried inside of B-80 is a "dictionary" of words that it can understand. Whenever you type something on the keyboard and then press the RETURN key, B-80 goes to that dictionary and looks for a match between what you just entered and a word in its dictionary. If it doesn't find a *perfect* match, it will issue a syntax error message.

B-80 is totally unforgiving about strange or misspelled commands, and the syntax error message is its way of telling you it doesn't understand what you want it to do. In fact, one of the first things you need to learn as a beginning programmer is what words do make sense to B-80.

Now let's try a word that B-80 does understand. Try typing in

```
FILES
```

and press the RETURN key. What happened? You should see all of the files that are on drive A. In other words, the B-80 direct mode command FILES does the same thing as the CP/M command DIR. We now know that FILES is part of the dictionary of words that B-80 understands.

Let's try another variation of the FILES command:

```
FILES "B:*.*"
```

and press the RETURN key. You should now see all of the disk files on drive B listed on the screen. As before, FILES is asking for a list of all disk files regardless of filename and filetype, but for the contents of drive B.

Now try typing in your name. Once again, unless you've got an unusual name, you will get a syntax error message, since it's unlikely that your name is in B-80's dictionary of words it understands.

Keyword. A **keyword** is a word that B-80 understands and causes some specific action to take place. It follows that keywords make up the dictionary of words that is embodied within B-80.

PRINT. Another keyword that is found in the B-80 dictionary is the PRINT command. Try typing in the following command:

```
PRINT 10
```

Don't forget to press the RETURN key. *The RETURN key is used by B-80 to signal that we have finished entering data from the keyboard and that it can begin using, or processing, whatever it was we just entered.* If you forget to press the RETURN key, the computer will obediently wait (forever?) for you to press the RETURN key. *We shall assume from now on that it is understood that RETURN must be pressed whenever we wish to enter data into the computer after it is typed in on the keyboard.*

What about adding two numbers together? For example,

```
PRINT 4+6
 10
Ok
```

prints 10 for the sum and then the "Ok" message to inform us that we are back in the direct mode of Basic again. Now try printing the answer to a subtraction problem. Did it work?

Now try printing your name. For example,

```
PRINT JACK
Syntax error
Ok
```

What went wrong? B-80 didn't understand what we were trying to do. *Whenever we want to print characters rather than numbers on the screen, the characters to be printed must be surrounded by quotation marks.* With that in mind, let's try it again:

```
PRINT "JACK"
JACK
Ok
```

Whenever we want to print a sequence (or "string") of characters in the direct mode, B-80 will insist that a quotation mark be placed before the first character and after the last character to be printed.